专 题

一种高效的服务组合优化算法

李俊, 郑小林*, 陈松涛, 陈德人

浙江大学计算机科学与技术学院, 杭州 310027

* 通信作者. E-mail: xlzheng@zju.edu.cn

收稿日期: 2011-08-29; 接受日期: 2012-01-06

国家自然科学基金 (批准号: 61003254)、浙江省自然科学基金 (批准号: Y1080130) 和中央高校基本科研业务专项资助项目

摘要 随着功能性属性相同而非功能性属性各异的 Web 服务的大量涌现, 如何在服务组合业务流程中为各个任务选择相应的组件服务以达到组合服务的 QoS (quality of service) 最大化, 并在此基础上满足不同用户的需求, 已成为了国内外研究的热点. 由于该问题的复杂性 (NP-hard), 目前存在的大多数方法都并不十分适合需要相对精确、实时决策的 Web 服务组合系统. 因此, 本文提出了一种基于凸包构建的组合服务优化算法 (CM-HEU) 用以解决 QoS 感知的服务组合优化问题. CM-HEU 首先通过对组合服务中的每组任务进行凸包构建, 以减少搜索空间. 然后通过对初始解向量的多次升级和一次降级操作以达到全局优化的目标. 实验表明: 相对于现阶段存在的一些主流方法, CM-HEU 不仅能得到一个比较理想的结果, 并且具有良好的效率.

关键词 服务组合 服务质量 多目标优化 用户需求 凸包

1 引言

随着 Web 服务技术的发展, 多个功能单一的 Web 服务能被灵活地、动态地组合成增值的并能满足不同用户需求的组合服务. 而整个服务组合过程可分为组合策划阶段和优化选择阶段, 组合策划阶段主要根据用户所提交的功能性需求来自动生成组合服务并确保组合流程的正确性, 目前对该阶段的研究主要有基于人工智能和图理论等方法^[1,2]. 优化选择的目的是为了满足用户对组合服务的非功能性需求, 即服务质量 (quality of service, QoS) 需求, 并使得最终组合服务的非功能性属性最大化; 该阶段需要为策划好的组合服务中, 由众多候选服务构成的各个任务选择一个服务, 以达到全局 QoS 的最大化并满足用户对服务质量的需求^[3]. 随着越来越多具有相同功能性属性而非功能属性各异的 Web 服务的涌现, 学术界关于对服务组合的研究正逐渐集中在如何进行 Web 服务的优化选择这一问题上.

由于 Web 服务的优化选择是一个被证明了的 NP 难问题^[3], 因此在数据量较大的情况下, 用枚举法不可能在合理的时间内得到最优的结果. 针对这一问题, 不同学者也提出了不同的解决方案. 如文献 [4] 中提出了一种基于整数规划的方法以获得 QoS 最优的组合服务, 但随着服务规模的增大, 该方法的时间复杂度呈指数级增加, 因此并不适合规模庞大的 Web 服务应用程序. 同样, 如文献 [5] 中提出的多目标遗传算法、文献 [6] 提出的混合枚举方法都无法从根本上解决耗时巨大的问题. 另外, 文献 [7] 提出了一种启发式算法 (WS-HEU), 该方法类似于用来解决多背包多选择问题的 M-HEU 算法^[8], 其

表 1 系统概念与参数
Table 1 System notions and parameters

S	a specific composition service
T_i	the i th task of S
s_{ij}	the j th candidate component service in T_i
Q_{ij}	QoS vector for s_{ij}
C	globe constrain vector
$\mathcal{F}(s_{ij})$	utility function for s_{ij}
$\mathcal{F}(S)$	utility function for S

时间复杂度为多项式时间. 但是, 仍然无法满足大多数 Web 服务应用程序的实时响应需求. 值得一提的是, 文献 [9] 提出了一种时间复杂度为线性时间的 C-HEU 算法, 但该算法也不适合服务质量属性值与最优结果密切关联的 Web 服务组合应用程序, 并且该算法在候选服务数量较大的情况下, 效果并不理想.

本文提出了一种针对服务优化选择的基于凸包构建的启发式算法 (CM-HEU). 该方法首先对构成组合服务流程的各个任务进行凸包构建, 这在有效地保留可能出现在优化结果中的服务的同时显著地减少了相应的搜索空间; 然后通过对初始解向量的多次升级和一次降级操作以达到全局优化的目标. 实验表明: 相对于现阶段存在的一些主流方法, CM-HEU 不仅能得到一个比较理想的结果, 并且具有良好的效率.

2 系统模型

2.1 系统概念

这一小节首先简要介绍本文中所需要涉及的一些概念及其参数如表 1 所示.

- 组合服务 $S = \{T_1, T_2, \dots, T_n\}$, 其中 $T_i (0 < i \leq n)$ 表示组合服务中的第 i 个任务, $n \in \mathbb{Z}^+$ 表示构建该组合服务所需的任务数.

- 组件服务 $s_{ij} (1 \leq i \leq n, 1 \leq j \leq l_i)$ 表示某一个能完成任务的组件服务. 其中 l_i 表示任务 T_i 中的候选服务数.

- 服务质量 $Q_{ij} = \{q_{ij1}, q_{ij2}, \dots, q_{ijh}\}$ 表示组件服务 s_{ij} 的 QoS 向量, 其中 h 表示相关 QoS 的属性数, 每一个属性有一个对应的相关领域值 v_{ijk} , 其中 $1 \leq k \leq h$.

- 需求向量 $C = \{C_1, C_2, \dots, C_m\}$ 表示用户对组合服务的全局 QoS 向量, 其中 $C_r (0 < r \leq m)$ 表示用户对第 r 个 QoS 属性的需求.

- 效用函数 $\mathcal{F}(s_{ij}) = \mathcal{G}^a(\psi_{ijk})$ 表示组件服务 s_{ij} 的效用函数, 其中 $\psi_{ijk} = \{\langle q, v \rangle | q \in q_{ijk}, v \in v_{ijk}\}$ 表示第 k 个属性上的二元函数, $\mathcal{G}^a$ 表示一个效用聚合函数.

- 效用函数 $\mathcal{F}(S) = \sum_{i=1}^n \sum_{j=1}^{l_i} x_{ij} \mathcal{F}(s_{ij})$ 表示组合服务 S 的效用函数, 即为目标函数. 其中 $x_{ij} \in \{0, 1\}$ 表示服务选择算子, 当 $x_{ij} = 1$ 时表示在第 i 个任务中选择了第 j 个服务, 并且 $\sum_{j=1}^{l_i} x_{ij} = 1$ 表示一组任务内只能有一个组件服务被选择.

2.2 QoS 模型

为了区分大量功能性属性相同的 Web 服务, 我们借助 Web 服务的质量指标来建立 Web 服务的

表 2 4 种 QoS 在顺序结构下的聚合函数

Table 2 Four QoS aggregation functions according to sequence composition pattern

QoS attributes	Aggregation function
Price (PR)	$PR(S) = \sum_{i=1}^n \sum_{j=1}^{l_i} x_{ij} PR(s_{ij})$
Response time (RT)	$RT(S) = \sum_{i=1}^n \sum_{j=1}^{l_i} x_{ij} RT(s_{ij})$
Availability (AV)	$AV(S) = \prod_{i=1}^n \sum_{j=1}^{l_i} x_{ij} AV(s_{ij})$
Reputation (RP)	$RP(S) = \prod_{i=1}^n \sum_{j=1}^{l_i} x_{ij} RP(s_{ij})$

QoS 模型. 为了便于 QoS 模型的建立和求解, 本文的 QoS 模型主要包含了 4 项通用指标^[10]. 因此, 一个组件服务的 QoS 属性可由一个 4 元组表示如下:

$$QoS(s_{ij}) = \langle RT(s_{ij}), PR(s_{ij}), AV(s_{ij}), RP(s_{ij}) \rangle, \quad (1)$$

其中每一个服务质量标准都有相应的计算和评价方法:

- 响应时间 (RT): 表示一个服务从用户发送请求开始到返回结果的平均时间. 包括服务处理时间 T_{pr} 和传输时间 T_{tr} . 总的响应时间为 $RT(s_{ij}) = T_{pr}(s_{ij}) + T_{tr}(s_{ij})$.
- 服务费用 (PR): 表示用户每次调用该服务所要支付的费用.
- 可用性 (AV): 表示一个服务正常被调用的概率. 由于 Web 服务是按次数进行调用的, 因此可由如下公式描述:

$$AV(s_{ij}) = \frac{\sum_{i=1}^n \lambda(s_{ij})_i}{n}, \quad (2)$$

其中 $\lambda(s_{ij})_i = 1$ 表示该服务第 i 次被正常调用, 否则 $\lambda(s_{ij})_i = 0$. n 表示该服务总的被调用次数.

- 声誉 (RP): 表示 Web 服务的可信度, 声誉越好意味着该 Web 服务的可信度越高, 即该服务的执行结果与其所描述的越接近. 它主要依赖于用户使用该服务的实际经验, 本文使用用户的平均经验信息来表示声誉, 可描述如下:

$$RP(s_{ij}) = \frac{\sum_{x \in Z} DT_x^s}{|Z|}, \quad (3)$$

其中 $|Z|$ 表示与服务有直接交互的请求者数量, DT_x^s 表示请求者 x 对 Web 服务的直接信任值, 可由如下公式描述:

$$DT_x^s = \sum_{\rho \in \kappa} \frac{\mathbf{R} \cdot \mathbf{N}}{\sigma \sum_{\ell=1}^k |N_\ell|} f_w(\rho) \quad (4)$$

其中 $\mathbf{R}$ 和 $\mathbf{N}$ 示请求者的需求向量与评价向量, 算子 $\cdot$ 表示向量的点积, σ 表示规约因子, 它将直接信任值规约到 0, 1 的范围内. $f_w(\rho) = \text{decay}(\rho) / \sum_{\rho \in \kappa} \text{decay}(\rho)$ 表示第 ρ 次交易的时间权重函数, 时间延迟因子为 $\text{decay}(\rho) = e^{\frac{-[t(n)-t(\rho)]}{D}}$, $t(n)$ 表示当前时间, $t(\rho)$ 表示表示发生第 ρ 次交互的时间.

QoS 指标包含正向指标和负向指标, 正向指标表示该指标的值越大越好 (如声誉、可用性等), 而负向指标表示该指标的值越小越好 (如响应时间、费用等). 有了组件服务的 QoS 指标及其相应的计算方法, 便可以按照一定的组合结构^[6]来计算组合服务的 QoS. 为了便于阐述, 本文只考虑其中的顺序结构. 表 2 显示了上述 4 个 QoS 属性的聚合函数. 其中 $x_{ij} \in \{0, 1\}$ 表示服务选择算子, 当 $x_{ij} = 1$ 时表示在第 i 个任务中选择了第 j 个服务, 并且 $\sum_{j=1}^{l_i} x_{ij} = 1$ 表示一组任务内只能有一个组件服务被选择.

2.3 目标函数

组合优化的目标是使得组合服务 S 的非功能性属性值最大化^[8]. 根据这一思想, 本文首先使用效用函数来表示单个组件服务的聚合 QoS 值. 再进一步综合各单个组件服务的效用函数以得到表示组合服务 QoS 效用的目标函数. 单个服务的效用函数可多准则加权决策模型^[11] 表述如下:

$$\mathcal{F}(s_{ij}) = \sum_{k=1}^h \omega_k q'_{ijk}, \quad (5)$$

其中 q'_{ijk} 为规范化后的 QoS 属性值. 规因为 QoS 属性值可以为正向属性, 如可用性 (availability) 属性, 其值越大表示质量越高; 也可以为负向属性, 如响应时间 (response time), 其值越大表示质量越低. 因此, 需要进行范化处理以使属性值在同一范围内操作. 其规范化处理过程可由如下公式表示:

$$q'_{ijk} = \begin{cases} \frac{q_{ijk} - q_{ijk}^{\min}}{q_{ijk}^{\max} - q_{ijk}^{\min}}, & \text{if } q_{ijk}^{\max} - q_{ijk}^{\min} \neq 0 \quad (\text{positive attributes}), \\ 1, & \text{if } q_{ijk}^{\max} - q_{ijk}^{\min} = 0, \end{cases} \quad (6)$$

$$q'_{ijk} = \begin{cases} \frac{q_{ijk}^{\max} - q_{ijk}}{q_{ijk}^{\max} - q_{ijk}^{\min}}, & \text{if } q_{ijk}^{\max} - q_{ijk}^{\min} \neq 0 \quad (\text{negative attributes}), \\ 1, & \text{if } q_{ijk}^{\max} - q_{ijk}^{\min} = 0, \end{cases} \quad (7)$$

其中, $q_{ijk} = v_{ijk}$ 表示组件服务 s_{ij} 在第 k 个 QoS 属性上的值. $q_{ijk}^{\min} = \min_{s_{ij} \in T_i} (q_{ijk})$ 表示任务 T_i 中所有组件服务在第 k 个属性上的最小值, 同样 $q_{ijk}^{\max} = \max_{s_{ij} \in T_i} (q_{ijk})$ 表示任务 T_i 中所有组件服务在第 k 个属性上的最大值. 综上所述, 可得组合服务的效用函数, 即优化的目标函数. 由于只考虑的组合结构中的顺序结构 (其他结构可以通过一定的技术转化为顺序结构), 因此采用简单的求和运算来得到目标函数, 形式化描述如下, 其中, $\omega_k > 0$ 且 $\sum_{k=1}^n \omega_k = 1$ 表示组合服务中每个任务的权重.

$$\mathcal{F}(S) = \sum_{i=1}^n \omega_k \mathcal{F}(s_{ij}). \quad (8)$$

2.4 问题描述

服务组合优化问题是如何在合理的时间内找到一个最好的, 即目标函数值最大的并且能满足各种用户 QoS 需求的组合服务而不用穷举出所有的可能性组合. 有了前面几节的基础, 本文将服务组合优化问题形式化描述如下:

对于一个给定的组合服务 $S = \{T_1, T_2, \dots, T_n\}$ 和 m 个全局 QoS 限制 $C = \{C_1, C_2, \dots, C_m\}$, 优化的目标是找到一个可能性的组合服务 S 使得下面公式成立.

$$\begin{aligned} & \text{Maximize } \mathcal{F}(S) \\ & \text{subject to } \sum_{i=1}^n \sum_{j=1}^{l_i} x_{ij} q_k(s_{ij}) \leq C_k (k = 1, \dots, m), \\ & \sum_{j=1}^{l_i} x_{ij} = 1, i = 1, 2, \dots, n, j = 1, \dots, l_i, \\ & x_{ij} \in \{0, 1\}, i = 1, \dots, n, j = 1, \dots, l_i, \end{aligned} \quad (9)$$

其中当 $x_{ij} = 1$ 时表示在第 i 个任务中选择了第 j 个候选服务, 否则 $x_{ij} = 0$. $q_k(s_{ij})$ 表示组件服务的 QoS 资源需要, 它的和必须满足全局 QoS 限制).

3 所提出解决方案

为了解决上述问题, 本文在 M-HEU^[8] 的基础上提出了一种启发式算法, 取名为 CM-HEU. 该算法的主要思想是将每组任务内的所有候选服务投影成二维空间上的点, x 轴坐标表示组件服务的总 QoS 资源需要, y 轴坐标表示组件服务的效用函数值. 然后在这些点集合的基础上构建凸包, 保留凸包上的所有顶点作为下一步的候选服务, 而摒弃所有其他的点. 最后, 基于 M-HEU 得到一个近似最优结果. 这样做主要是根据文献 [12] 中所提的“统治点”这一概念: 一些相对资源消耗大并且效用低的结点永远不会在最优结果中出现. 因此, 我们综合了凸包的性质, 仅仅将一些最可能出现在最优结果中的结点保存下来. 实验表明, 尽管这样处理也会丢掉一些可能出现在最优结果中的结点, 但对最后的结果影响并不大, 并且能节省大量的时间, 特别适用于需要实时响应的服务组合应用系统. 该算法的具体步骤可描述如下:

- 步骤 1: 构建凸包.

1.1 使用如下公式, 将各组任务中所有的候选服务的 QoS 向量转化为一个标量;

$$a_{ij} = \frac{q_{ij} \cdot C}{|C|}, \quad (10)$$

其中 a_{ij} 表示组件服务 s_{ij} 的总 QoS 资源需求值, q_{ij} 为组件服务 s_{ij} 的 QoS 向量, C 为全局 QoS 向量.

1.2 应用 Graham's 算法^[13] 对每个任务进行凸包的构建. 这里每个组件服务对应二维空间里的一点, a_{ij} 表示该点的横坐标, $\mathcal{F}(s_{ij})$ 表示该点的纵坐标.

1.3 返回凸包的各个顶点, 并添加到该任务的候选服务数组.

- 步骤 2: 寻找初始解.

2.1 将各个任务中的组件服务按效用值非递减排列, 并分别选择第 1 个服务作为初始解. 并得到当前总 QoS 消耗向量 $Q = \sum_{i=1}^n q_{ijk} \mathfrak{R}_i$, 其中 $\mathfrak{R}$ 表示当前解向量.

2.2 定义可行性因子 $f_\mu = Q_\mu / C_\mu = \max_{k \in m} Q_\mu / C_\mu$, 判断如果 $f_\mu \leq 1$, 则转到步骤 3.

2.3 定义总资源节省因子 $\Delta a_{ij} = (q_{i\mathfrak{R}_i} - q_{ij}) \times C / |C|$, 考虑在所有任务组中找一个具有最大 Δa_{ij} 的并能满足以下 3 个条件的候选服务来替换 $\mathfrak{R}_i$, 替换后的解向量记为 $\mathfrak{R}'_i$.

- 1) $Q_\mu(\mathfrak{R}') < Q_\mu(\mathfrak{R})$,
- 2) 如果 $k \neq \mu$ 并且 $Q_k(\mathfrak{R}) \leq C_k(\mathfrak{R})$, 则有 $Q_k(\mathfrak{R}') < C_k(\mathfrak{R}')$,
- 3) 如果 $k \neq \mu$ 并且 $Q_k(\mathfrak{R}) > C_k(\mathfrak{R})$, 则有 $Q_k(\mathfrak{R}') < C_k(\mathfrak{R})$.

判断如果能找到该组件服务, 则确定将该服务替换当前解向量中的服务并转到步骤 2.2, 否则找不到可行解, 并结束算法.

- 步骤 3: 升级解向量 (升级表示解向量的总效应值增加).

3.1 不断在所有任务组中找一个具有最大 Δa_{ij} 的候选服务来进行升级. 如果该升级可行, 则继续.

3.2 定义总资源消耗升级因子 $\Delta p_{ij} = (\mathcal{F}_{i\mathfrak{R}_i} - \mathcal{F}_{ij}) / \Delta a_{ij}$, 其中 $\mathcal{F}_{ij}$ 表示 s_{ij} 的效用值. 在所有任务组中找一个具有最大 Δp_{ij} 的候选服务来进行升级. 如果该升级可行, 则转到步骤 3.1.

- 步骤 4: 一次升级解向量并若干次降级解向量.

4.1 定义非理性升级因子 $\Delta p'_{ij} = (\mathcal{F}_{i\mathfrak{R}_i} - \mathcal{F}_{ij}) / \Delta t'_{ij}$, 其中 $\Delta t'_{ij} = \sum_{k=1}^m (q_{i\mathfrak{R}_i k} - q_{ijk}) / (C_k - Q_k)$. 在所有任务组中找一个具有最大 $\Delta p'_{ij}$ 的候选服务来进行升级.

4.2 定义降级因子 $\Delta p''_{ij} = (\mathcal{F}_{i\mathfrak{R}_i} - \mathcal{F}_{ij}) / \Delta t''_{ij}$, 其中 $\Delta t''_{ij} = \sum_{k=1}^m (q_{i\mathfrak{R}_i k} - q_{ijk}) / (Q_k - C_k)$. 在所有任务组中找一个具有最小的 $\Delta p''_{ij}$ 的候选服务来进行降级. 如果该服务能被找到, 并且当前解向量可行.

则转到步骤 3. 否则, 继续进行降级处理.

4.3 算法结束并返回当前解向量 $\mathcal{R}$.

该算法的时间复杂度为 $O(nlm + nllgl + n^2(l' - 1)^2m) = O(n^2(l' - 1)^2m)$, 为多项式时间. 值得注意的是, 当进行凸包构建后, 每组任务所返回的候选服务数量并不统一. 因此, 这里的为各组中的最大候选服务数.

4 实验分析

为了分析 CM-HEU 算法的性能, 本文利用 Java 语言实现了该算法, 并同时实现了现阶段主流的其他 5 种算法来进行分析和比较, 分别为:

- BBLP 算法^[14], 它利用迭代不断产生查找树来得到最优解, 它是一个最优算法. 目前存在的大多数最优算法都基于 BBLP. 该算法主要用于产生最优结果.

- Upper Bound 算法^[14], 它利用线性单纯性方法来找到目标函数的上界. 该上界的值比最优解的值要高.

- M-HEU^[8], 一种基于总资源消耗的启发式算法, 该算法用于解决 MMKP 问题.

- WS-HEU^[7], 一种用于解决 QoS 感知的 Web 服务组合优化问题的启发式方法.

- C-HEU^[9], 一种基于凸包构建的复杂度为线性时间的启发式算法, 与 CM-HEU 不同的是, 它将凸包构建后的点集在聚合在一个二维空间内, 并根据点的坐标进行排序和替换进而更新解向量. 该算法有个前提就是需求的资源和消耗的资源不是正比关系, 因此虽然它时间复杂度很低 (线性时间) 但不适合大规模的服务组合系统.

本文的实验环境是在 CPU 为 Intel Duo 2.4 GHz, 内存为 2 GB 的 ThinkPad T61 机器上进行的, 操作系统为 Windows XP, 并利用了 Eclipse 开发工具, 编程语言为 Java.

4.1 数据集与测试实例

本文主要利用了两种数据集来测试算法的性能表现. 第 1 种是名为 QWS 的真实数据集, 该数据集包含了 2507 个真实 Web 服务及 9 种 QoS 属性的测量值, 关于该数据集的具体信息可参见文献 [15,16]. 第 2 种为随机数据集, 该数据集集中有 10000 个 Web 服务并且其 QoS 值均匀分布在 0 和 1 之间. 产生随机数据的主要原因是为了更全面的反应算法的性能表现.

由于本文的多数实验是在大型数据集上进行的, 因此要得到 BBLP 算法的结果是不现实的 (运行一次的平均时间可能需要几天或几年). 因此, 本文利用 Upper Bound 方法计算得到目标函数的上界值作为最优结果. 优化率即为各个算法与上界的比值. 另外, 本文对每一组数据测试 10 次, 并取其平均值为最终结果.

4.2 结果分析

首先通过变化任务组数和候选服务数来观察 4 种启发式算法优化率的影响. 图 1 表示在各组的候选服务数 $l = 150$, QoS 属性数 $m = 9$, $n = 5 \sim 50$ 的情况下, 分别基于 QWS 数据集和随机数据集的各启发式算法优化率的性能表现. 结果表明 CM-HEU 的优化率明显要高与 C-HEU, 并且非常接近于

图 1 当 $l = 150$, $m = 9$, $n = 5 \sim 50$ 时, 各算法优化率的变化情况

Figure 1 The optimality of each algorithm when $l = 150$, $m = 9$, $n = 5 \sim 50$

图 2 当 $n = 20$, $m = 9$, $l = 50 \sim 500$ 时, 各算法优化率的变化情况

Figure 2 The optimality of each algorithm when $n = 150$, $m = 9$, $l = 50 \sim 500$

M-HEU, 甚至在 QWS 数据集中多数性能表现还高于 WS-HEU. 其中 C-HEU 表现最差是因为它在构造凸包的过程中完全摒弃了各组任务中的一些可能达到最优结果的服务, 并且在替换服务的过程中也可能忽略掉一些性能好的服务, 而 CM-HEU 仅仅是忽略一些最不可能出现在最优结果中的服务并且基于构建后的服务集合进行全局优化以得到理想的结果. 另外, 我们可以观察到, C-HEU 在随机数据集中的性能表现要优于 QWS 数据集, 这是因为 QWS 数据集与效用值的关联度要高于随机数据集. 这也证明了 C-HEU 并不适合数据紧密关联的服务组合系统. 从图 1 中也可以发现, 测试数据为随机数据集时, 各个算法的优化率性能随着 n 的增大而增加. 当 n 大于 50 时, CM-HEU 的优化率平均能达到 99.5% 左右. 同样, 在 QWS 数据集中, 也能达到较高的优化率, 约为 98% 左右.

图 2 表示在各组的候选服务数 $n = 20$, QoS 属性数 $m = 9$, $l = 50 \sim 500$ 的情况下, 分别基于 QWS 数据集和随机数据集的各启发式算法优化率的性能表现. 从图 2 中可以观察到, 随着 l 的变化, CM-HEU 也能达到较高的优化率: 在 QWS 数据集和随机数据集中的优化率性能表现分别约为 98% 和 96.5%. 明显高于 C-HEU. 另外, 由于 QWS 数据集的不规则性和初始解的构造方式不同导致了 CM-HEU 在多数情况下优化率性能高于 WS-HEU.

图 3 当 $l = 150, m = 9, n = 5 \sim 50$ 时, 各算法计算时间的变化情况

Figure 3 The computational time of each algorithm when $l = 150, m = 9, n = 5 \sim 50$

图 4 当 $n = 20, m = 9, l = 50 \sim 500$ 时, 各算法计算时间的变化情况

Figure 4 The computational time of each algorithm when $n = 20, m = 9, l = 50 \sim 500$

图 3 和 4 表示各算法随着数量增多的时间性能表现. 从图中可以明显的观察到, 不管是随机数据集还是 QWS 数据集, M-HEU 和 WS-HEU 的计算时间随着任务组数或候选服务数的增加而呈二次方增加. 而 C-HEU 和 CM-HEU 的计算时间则相对要小很多, 当 $n = 50, l = 150$ 时, CM-HEU 的计算时间约为 100 多毫秒而 M-HEU 和 WS-HEU 则需要约为 8000 多毫秒. 这是因为 CM-HEU 是基于凸包构造后的数据集上进行全局优化的, 从而显著地减少了搜索空间. 另外, 还可以观察到 CM-HEU 的计算时间会随着任务组数的增加而增加, 而候选服务数的增加对其则影响不大, 基本维持在几十毫秒左右. 这是因为每次构造凸包所返回的结点是一定的. 尽管 C-HEU 所花的时间代价最少, 但是它所得结果的优化率普遍偏低且不稳定. 其内在特性也表明了该算法不适合大规模的服务组合系统. 因此, 从权衡优化率和计算时间的角度来看, CM-HEU 更能达到所想要的效果, 较适合需要得到良好结果并要求实时响应的服务组合系统.

综上所述, CM-HEU 不仅能得到一个近似最优化的结果, 并且所需时间较为合理. 从权衡优化率和计算时间的角度看, CM-HEU 都比其他几种启发式算法更优越.

5 结论

本文为解决服务质量感知 Web 服务组合问题提出了一种基于凸包构建的启发式方式, 实验结果表明该方法能在合理的时间内找到一个近似最优的结果. 并且从权衡优化率和计算时间的角度看, 比其他几种主流的启发式算法更优越. 由于 Web 服务组合系统对组合 QoS 结果的精确度要求不是最高, 但较高, 并且对相应时间要求也相对较高. 因此, 对于大规模的服务系统来说我们相信 CM-HEU 会是一个不错的选择.

考虑到目前一些服务组合优化方法都是在假定 QoS 信息可信的基础上提出的, 而在面向服务的环境中, 很有可能存在欺诈、提供虚假信息等恶意行为, 因此也需要一种机制来判断 QoS 信息的可信性. 下一步的工作中, 我们将考虑结合服务的信任和信誉问题, 研究一种可信的、高效的服务组合优化机制.

参考文献

- 1 Li L, Liu D X, Bouguettaya A. Semantic based aspect-oriented programming for context-aware Web service composition. *Inf Sys*, 2011, 36: 551-564
- 2 Milanovic N, Malek M. Search strategies for automatic web service composition. *Int J Web Serv Res*, 2006, 3: 1-32
- 3 Pisinger D. Algorithms for Knapsack Problems. Dissertation for Doctoral Degree. Denmark: University of Copenhagen, 1995
- 4 Zeng L Z, Benatallah B, Ngu A H H, et al. QoS-aware middleware for web services composition. *IEEE Trans Softw Eng*, 2006, 30: 311-327
- 5 Wada H, Suzuki J, Yamano Y, et al. A multi-objective optimization framework for SLA-aware service composition. *IEEE Trans Serv Comput*, 2011, 99: 1-14
- 6 Qi L Y, Tang Y, Dou W C, et al. Combining local optimization and enumeration for QoS-aware Web service composition. In: *Proceedings of IEEE International Conference on Web Services*, Miami, 2010. 34-41
- 7 Yu T, Zhang Y, Lin K J. Efficient algorithms for web services selection with end-to-end QoS constraints. *ACM Trans Web*, 2007, 1: 1-26
- 8 Akbar M M, Manning E G, Shoja G C, et al. Heuristic solutions for the multiple-choice multi-dimension knapsack problem. In: *Proceedings of the International Conference on Computational Science*, London, 2001. 659-668
- 9 Akbar M M, Rahman M S, Kaykobad M. Solving the multi-dimensional multiple-choice knapsack problem by constructing convex hulls. *Comput Operat Res*, 2006, 33: 1259-1273
- 10 Zaki M, Athman B. Trust Management for Service-Oriented Environments. Heidelberg: Springer, 2009
- 11 Hwang C L, Yoon K. Multiple Attribute Decision Making: Methods and Applications. Heidelberg: Springer-Verlag, 1981
- 12 Kellerer H, Pferschy U, Pisinger D. Knapsack Problems. Heidelberg: Springer, 2004
- 13 Thomas H C, Charles E L, Ronald L R, et al. Introduction to Algorithms. Beijing: China Machine Press, 2006
- 14 Khan S. Quality Adaptation in a Multisession Multimedia System: Model, Algorithms and Architecture. Dissertation for Doctoral Degree. Victoria: University of Victoria, 1998
- 15 Al-Masri E, Mahmoud Q H. The QWS dataset. [Http: //www.uoguelph.ca/ qmahmoud/qws/index.html/](http://www.uoguelph.ca/qmahmoud/qws/index.html/)
- 16 Al-Masri E, Mahmoud Q H. Investigating web services on the world wide web. In: *Proceedings of the International World Wide Web Conference*, Beijing, 2008. 795-804

An efficient algorithm for QoS-aware service composition

LI Jun, ZHENG XiaoLin*, CHEN SongTao & CHEN DeRen

College of Computer Science, Zhejiang University, Hangzhou 315100, China

*E-mail: xlzheng@zju.edu.cn

Abstract With the rapid increasing number of the independently developed web services that provide the same functionality but differ in quality of service (QoS), how to select corresponding component services for each task of a specific service composition process in the way that maximizes the aggregation QoS value, while satisfying the constraints set by the end-users is practice known as an NP hard problem. Current solutions are not suitable for most real-time decision-making service composition applications which are required to obtain an optimal result in reasonable time. In this paper, we present an approach which addresses the issue of this problem by constructing convex hulls. The proposed heuristic algorithm consists of two steps: first, convex hulls are constructed for reducing the search space, then the multi-objective optimization problem is solved based on M-HEU. The results of the experiments demonstrate that our approach can obtain a close-to-optimal solution in a reasonable amount of time.

Keywords service composition, quality of service, multi-objective optimal, user constraints, convex hulls

LI Jun was born in 1982. He received the M.S. degree in computer science from Zhejiang Gongshang University, Hangzhou in 2004. Currently, he is a Ph.D. candidate at College of Computer Science of Zhejiang University. His research interests include service computing and trust computing.

ZHENG XiaoLin was born in 1977. He received the Ph.D. degree in computer science from Zhejiang University, Hangzhou in 2004. Currently, he is an Associate Professor at College of Computer Science of Zhejiang University. His research interests include service computing, social computing, opinion mining. Dr. Zheng is a senior member of CCF, also a member of ACM and IEEE, and the Director of the Modern Services Laboratory in Zhejiang University.

CHEN SongTao was born in 1987. He received the B.S. degree in computer science and technology from Qufu University, Rizhao in 2010. Currently, he is a graduate student at Zhejiang University. His research interests include services computing, social networks, and trust computing.

CHEN DeRen was born in 1951. He received the M.S. degree in computer science and applied mathematics from Zhejiang University, Hangzhou in 2004. Currently, he is a Professor at College of Computer Science of Zhejiang University. His research interests include service computing and electronic commerce. Prof. Chen is an editorial board member of International Journal of Web Service Research (JWSR), and the Director of E-Service Research Center in Zhejiang University.