第 38 卷 第 3 期
Vol.38 No.3

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

• 软件技术与数据库 •　　文章编号：1000—3428(2012)03—0039—04　　文献标识码：A　　中图分类号：TP311

一种软件工作量估算的不确定性度量方法

解 浪[1,2]，杨 叶[2]

(1. 中国科学院研究生院，北京 100190；2. 中国科学院软件研究所互联网软件技术实验室，北京 100190)

摘 要：以 COCOMO81 模型为基础，结合模型输出方差以及模型与数据自身方差的组合，分别度量模型本身估算值与实际值差距的不确定性，并在此基础上给出预测区间。提出以命中率(SR)与平均相对宽度(MRV)相结合的评测标准。通过采用重采样获得 *N* 对训练和测试集合，计算不同置信度下区间 SR 和 MRV 的均值，运用 SR 和 MRV 的散点图比较不同度量方法获得区间。实验结果表明，该方法能以相同的命中率获得更窄的区间。

关键词：不确定性；软件工作量估算；重采样；预测区间；COCOMO 模型；方差

Uncertainty Measurement Method of Software Effort Estimation

XIE Lang[1,2], YANG Ye[2]

(1. Graduate University of Chinese Academy of Sciences, Beijing 100190, China;
2. Lab for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

【Abstract】This paper proposes a method to assess the uncertainty of using COCOMO81 model. It uses the variance of model's output to assess the uncertainty of model's output. And uses both the data's variance and the model's output to assess the uncertainty of the distance between estimated effort and its real value. Based on the uncertainty, it proposes the prediction interval. It proposes the Mean Relative Variance (MRV) and shooting ration as the measurement to assess the uncertainty. In the experiment, re-Sampling method is employed to get N pairs of train set and test set. The average value of SR and MRV for different confidence level is calculated. After that, the scatter plot of SR and MRV is used to compare the different intervals that generated by different methods of assessing uncertainty. Experimental result shows that the measurement for uncertainty can generate narrower interval at the context of the same shooting ratio.

【Key words】uncertainty; software effort estimation; re-sampling; prediction interval; COCOMO model; variance

DOI: 10.3969/j.issn.1000-3428.2012.03.014

1 概述

软件项目结束前，在任何环节制定和调整项目计划时，就必须对项目目前进度所需要的人力及其他资源、项目持续时间和项目成本做出估算。成本估算和成本管理是软件项目管理的核心任务之一。软件成本的单位主要使用“人/月”。主要的成本估算方法分为：基于算法模型，基于非算法模型和组合方法[1]。3 类方法的共同点均是以历史数据为基础，根据当前项目、当前进进度中能搜集到的变量数据来预测成本。项目数据中的一个重要因素是软件规模，通常规模的度量主要分代码行和功能点[2]2 类方法。

历史数据存在不确定性，主要表现在以下 2 点：(1)除成本外的其他变量，比如 COCOMO 数据的驱动因子，在搜集过程中对估算方法的理解以及评分，除人为造成的偏差外，同时存在着随机性(也称不确定性，本文中将不加区别)，同时这也是数值上表征为方差的部分。(2)项目整个过程中的任何具有影响的随机事件(如开发人员某段时间的状态)都会导致最终的成本出现随机性。这 2 类历史数据的不确定性的存在，导致了估算结果的不确定性。而当前项目的不确定因素也导致了当前项目的实际成本具有一定不确定性。在学术研究中，这些数据的不确定性导致了评测一个方法的指标具有不确定性，同时导致了实际应用中不确定性估计不足，这是软件成本超支严重的重要原因。

软件成本估算的不确定性研究是成本估算领域具有挑战性和重要价值的问题。本文在 COCOMO 模型的基础上提出预测区间，使用成本估算区间代替原先的估算值，一定程度上缓解了不确定性的问题。

2 相关工作

2.1 COCOMO81 模型与数据

COCOMO(Constructive Cost Model)是 Boelm 教授提出基于算法模型的估算方法，使用回归方法获得模型参数。其主要有 2 个版本：COCOMO81(C81)[3]和 COCOMOII(CII)[4]。C81 于 1981 年发布，有公开的数据，本文的实验是使用最早用来校准 C81 模型的数据和搜集于 NASA 6 个组织的 93 调数据[5]。分别表示为 C-81(加横扛区别于模型 C81)和 NASA93。C81 模型拥有 3 个等级模型[3]：基本(basic)模型，中等(mediate)模型和详细(detail)模型。这 3 个模型对应于软件开发的不同状态。C81 的 3 个等级模型均满足下式：

$$effort = a \times KDSL^b \times F \tag{1}$$

其中，*effort* 为工作量，单位为人/月；*a* 和 *b* 为系数，具体的值取决于建模等级(基本、中等或详细)以及项目的模式(组织型、半独立型或嵌入型)这个系数的取值由专家意见来决定先验值，然后结合 C-81 的 63 条数据进行校准；*KDSI* 为软件项目开发中交付的源指令(Delivered Source Instruction, DSI)千

基金项目：国家自然科学基金资助项目(90718042)；广东省中国科学院全面战略合作基金资助项目(2009B091300131)

作者简介：解 浪(1986－)，男，硕士研究生，主研方向：机器学习，成本估算；杨 叶，副研究员、博士

收稿日期：2011-08-12　**E-mail：**xielang@itechs.iscas.ac.cn

行数，也有用代码行(Line of Codes, LOC)表示，代表着软件规模(size)；F 是调整因子，基本模型 F=1。在中级和详细 2 个模型中，F 为 15 个成本因子对应乘数的乘积：$\prod_{i=1}^{15} EM_i$ 。实际使用中通常是利用局部数据进行本地校准[3-4]，校准过程则是对式(1)对数变换后，使用线性回归获得到 ln(a)和 b：

$$\ln(effort)-\ln(F)=\ln(a)+b\times\ln(KDSL) \tag{2}$$

为适应越来越复杂的软件系统，Boelm 教授于 2000 年提出了 COCOMOII 模型，由于数据保密等原因，本文暂不涉 COCOMOII 模型。

2.2 不确定性的度量

基于对度量成本估算的不确定性方面工作的调研[4,6-9]，已有的方法可分为以下 3 类方法：

(1)通过使用模型在历史数据上的 Magnitude Relative Erro(MRE)或者 Relative Erro(RE)的直方图来直接截取部分，作为估算值与实际值差值的范围的依据，计算出区间[4,6]。这种方法的优势在于对数据噪音不敏感，MRE 或者 RE 的区间在训练数据上直接确定后，不再变动，预测时与新数据的特征没有任何关系。如在文献[4]中，对于 MRE 集合进行截取，排序后的 MRE，取第 n%的值，作为估算区间的依据，带入 MRE 计算公式以及估算值计算出 2 个值作为估算区间。这里理想假设认为实际值有至少 n%的可能性落在该区间内。在文献[6]中通过假设 MRE 或者 RE 满足一定分布，使用 MRE 和 RE 的方差和期望的估算值来获得 MRE 和 RE 相应的百分位，从而计算相应置信度得区间，同样也是假设实际值至少以同样置信度落在该区间内。但是这种方法往往数据信息利用不充分，造成区间在局部过大或者过小，不仅影响精确性(实际值是否落在该区间内)，而且会对实用性产生影响(区间过大导致估算过高，资源分配浪费)，极端情况就是估算区间无穷大，虽然能保证一定能包含实际值，但是没有任何实际价值。

(2)对 MRE 或者 RE 的分布建立模型，输入与估算方法相同，并在同一份数据上校准新的模型。预测时，对新数据使用估算模型估算出成本，使用新模型估算出 MRE 或者 RE 的上下界，从而获得成本的上下界[7-8]。以 MRE 或者 RE 为输出，软件项目数据(不包括成本这一项)为输入的模型范围非常广，文献[7]使用线性模型，使用软件项目数据的部分特征和当前估算模型在训练数据上的 MRE 或者 RE，来回归新模型的参数；文献[8]利用神经网络，使用软件项目数据所有特征作为输入，而利用估算模型在训练数据上得到 MRE 和 RE，进行归一化后作为输出，训练神经元参数。在预测时，两者均是使用估算模型获得成本估算值，在使用新模型估算出 MRE 或者 RE，从而计算出估算区间。这种方法的优势在于充分利用数据中的信息，但是容易造成过拟合，另外可解释性差，没有将原有的估算模型的特点考虑进来，2 个模型没有必然联系，在分析偏差和方差时则造成了不便。

(3)直接使用专家知识来给出估算结果的不确定性[9]。这种方法通过搜集多个专家判断，最终给出在当前估算值下的不确定性，从而获得一个估算区间。优势在于人为因素加入，对数据噪音极其不敏感。劣势在于需要领域专家，移植性不好，代价也大。

3 基于 COCOMO 的不确定性度量方法

3.1 总体流程

本文方法包括使用历史数据训练和预测两块，整个流程表示如图 1 所示。

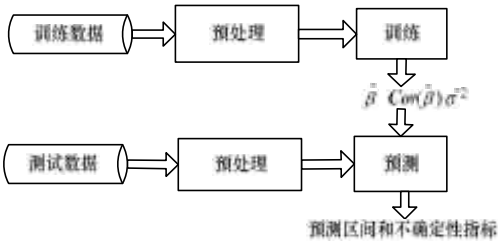


图 1 本文方法总体流程

预处理部分则是把数据按照式(2)进行整理，为方便表示，用 y 表示 ln(*effort*)，$\boldsymbol{x}$ 表示变换后的 ln(*size*)和常数 1 构成的向量。一般有如下基本假设：$y=\boldsymbol{\beta}^{\mathrm{T}}+\varepsilon$，其中，$\varepsilon\sim N(0,\sigma^2)$ 表示方差为 σ^2、期望为 0 的正态分布。

模型参数求解(训练)就是简单的线性回归。训练部分得到 3 个参数：$\hat{\beta}$ 是回归的系数，在本文中对应于 a 和 b；$\hat{\boldsymbol{Cov}}(\hat{\beta})$ 是 $\hat{\beta}$ 的协方差矩阵；$\hat{\sigma}^2$ 是训练数据获得的 σ^2 的有偏估计：均方误差(MSE)。这些均是在预测阶段获得区间的重要参数，将在 3.2 节中详细说明它们的计算和意义。在模型参数获得基础上很容易计算出对数变换后的目标方差和期望值。

在预测部分，由式(1)和式(2)可以看出，COCOMO 的估算值是在线性表达式上加了指数运算，这导致了密度函数的变形，直接推导方差非常困难[10]。其次充分利用指数函数的单调性。由模型 3 个参数，很容易获得线性模型输出 y 的方差和期望值。在固定置信度以及置信区间的计算方式，获得当前 y 的上下界，从而带入指数函数得到预测区间的上下界。最后通过预测区间中引出最终不确定性的度量指标。

3.2 模型参数的求解

通常使用零回归代替普通的最小二乘，防止矩阵退化[10]。而回归是特定分布下的最大似然，通过极大化似然函数即可获得。获得 $\hat{\beta}$，$\hat{\sigma}^2$ 和 $\hat{\boldsymbol{Cov}}(\hat{\beta})$ 计算公式如下：

$$\hat{\beta}=(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}+\lambda I)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y} \tag{3}$$

$$\hat{\sigma}^2=\frac{1}{N}(\boldsymbol{Y}-\boldsymbol{X}\hat{\beta})^{\mathrm{T}}(\boldsymbol{Y}-\boldsymbol{X}\beta) \tag{4}$$

$$\hat{\boldsymbol{Cov}}(\hat{\beta})\approx\hat{\sigma}^2(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}+\lambda I)^{-1} \tag{5}$$

其中，$\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 分别为训练数据转换后的数据矩阵和目标向量；N 为训练数据的数量；λ 为岭回归参数。式(5)由最小二乘回归与最大似然在特定分布下的等价性和最大似然性质[10]可推导获得。

3.3 线性回归问题中的 2 类不确定性方法

首先解释 2 个量：$\sigma_{\hat{y}}$ 和 $\sigma_{y-\hat{y}}$。在符号上沿用统计学的传统，加帽表示相应的变量样本估计值。$\sigma_{\hat{y}}$ 表示 $\hat{y}$ 的方差，而 $\sigma_{y-\hat{y}}$ 表示 $y-\hat{y}$ 的方差，这 2 个变量的意义不同，前者是单纯的线性模型预测值的方差，后者是预测值和实际值的差值的方差。这 2 个变量的估计值计算公式如下：

$$\hat{\sigma}_{\hat{y}}^2=\boldsymbol{x}^{\mathrm{T}}\hat{\boldsymbol{Cov}}(\hat{\beta})\boldsymbol{x} \tag{6}$$

$$\hat{\sigma}_{y-\hat{y}}^2=\sigma^2+\sigma_{\hat{y}}^2 \tag{7}$$

这里的 $\hat{\boldsymbol{Cov}}(\hat{\beta})$ 和 $\hat{\sigma}^2$ 就是通过式(4)和式(5)在训练数据上计算获得。这里的 x 是当前需要估算的项目数据，经过同样的预处理后获得。

从这 2 个量可以看出，最终估算值和实际值的差距的不确定性存在 2 个部分，一部分是模型输出的方差 $\sigma_{\hat{y}}^2$，由模型

参数的协方差带来，使用估算值 $\hat{\sigma}_{\hat{y}}^2$；另一部分是当前数据自身的方差 σ^2，由于对当前数据没有任何经验，因此使用了训练数据的 $\hat{\sigma}^2$ 代替。实质上是模型参数的协方差，从计算公式上 看，也是来源于训练数据，可以看做是历史数据的影响力在这个模型结构下的不同体现。

3.4 C81 的预测区间

首先计算 y 的区间。使用基于 t 分布的置信区间，一方面由于模型假设(式(2)的 epsilon 是正态分布)，另一方面由于方差本身的未知，使用的估算值，用 t 代替 z 分布[10]：

$$[\hat{y}-t_{d,1-\alpha}\times\sigma_{y-\hat{y}}, y+t_{d,1-\alpha}\times\sigma_{y-y}] \tag{8}$$

其中，t 的下标 d 表示自由度(由训练样本数量和 x 维度决定，维度减去 1)；$1-\alpha$ 表示置信度。通过查表获得对应的 t 值即可。

由于 *effort* 是 y 的函数：exp(y)、exp 函数的单调递增性，可以直接将 y 的上下界代入计算工作量的上下界。注意到一个问题，也是指数函数的增长速度，导致了分布随着 *effort* 增加而变稀疏，同时导致了工作量预测区间变大。区间变大的关键在于上界过大。因为背后的分布根本不知道，没有必要进行如此严格遵循的这些假设，使用一个小变换，使得上界控制在一个较小的范围之内：

$$effort_{\text{donw}} = F\times e^{\hat{y}-t_{d,1-\alpha}\times\sigma_{y-\hat{y}}} \tag{9}$$

$$effort_{\text{up}} = \min(effort_{\text{up1}}, effort_{\text{up2}}) \tag{10}$$

其中：

$$effort_{\text{up1}} = F\times e^{\hat{y}+t_{d,1-\alpha}\times\sigma_{y-\hat{y}}}$$

$$effort_{\text{up2}} = 2.5\times F\times e^{\hat{y}}+1.5\times effort_{\text{down}}$$

在式(9)和式(10)中，$effort_{\text{down}}$ 和 $effort_{\text{up}}$ 分别表示上界和下界。$effort_{\text{up2}}$ 实质上是用估算值加上估算值本身和下界差距的 1.5 倍。这个参数 1.5 是在实验中经验取值。在后面 C-81 和 NASA 数据集上的实验会看到适当的降低上界，并不会对命中率造成多大的影响。

3.5 C81 的不确定性度量

预测区间的提出对实际估算很有帮助，使用一个命中率高而且相对较窄的区间代替估算值是一个进步。在研究过程中需要衡量一个模型在使用过程中的不确定性，这需要一个指示量来衡量。使用区间的宽度的 0.5 倍，除以区间的中心作为不确定性的度量指标，表示为 RL(Relative Length)。在测试集上做平均，表示为 MRL(Mean RL)。计算方法如下：

$$RV = \frac{effort_{\text{up}}-effort_{\text{down}}}{effort_{\text{up}}+effort_{\text{down}}} \tag{11}$$

$$MRV = \sum_{i=1}^{M} RV_i \tag{12}$$

其中，M 为测试集的大小。这个指标有 2 个非常好的性质，就是值域在[0,1]和不依赖于项目成本的量级。

3.6 评测方法

不确定性指标源于区间，为对比区间的好坏进行评测。使用命中率(Shooting Ration, SR)和区间的相对宽度来对比区间。区间的相对宽度即本文提出的 RV。对于本文的区间就是使用 3.5 节中 RV 计算公式。命中率的定义源于命中，命中定义为实际值落在区间内，在一个测试集合上，实际值落在估算区间内的比例则是命中率。在下面实验中同时对比不同估算区间方法的命中率和区间相对宽度。

实验重复按一定比例随机切分数据集，获得一个集合，集合的元素是(训练集，测试集)对，在这些集对上进行实验，对各个指标最终在各个测试集上取平均值进行对比。这种方法适用于在软件估算领域数据量不足、重复可能性低的特点。

4 实验结果与分析

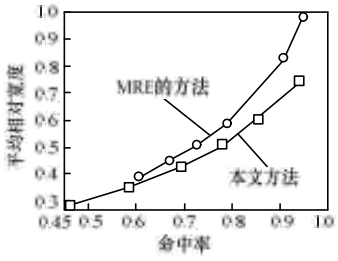
4.1 实验

实验中设置为：重采样次数为 2 000 次，训练集和测试集大小的比例为 3:1。在 C-81 和 NASA93 上本别校准 C81 模型，校准方法为回归求解参数 a 和 b。分别对比第 3 节中的方法和基于 MRE 的方法中结果最优的一种。基于 MRE 的方法的主要流程为：使用训练集合上的 MRE 集合，按增序排列，保留第 1-alpha 个(换算成数量取整)MRE 的值，使用该 MRE 值在训练集上结合成本估算值计算上下界。表 1 为本文方法和基于 MRE 方法的命中率和 MRV。

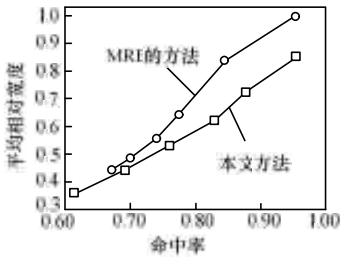
表 1 本文方法与 MRE 方法性能对比

方法	alpha 值	C-81		NASA93	
		命中率	MRV	命中率	MRV
MRE 方法	0.10	0.87	0.81	0.87	0.79
	0.15	0.81	0.58	0.83	0.64
	0.20	0.77	0.51	0.78	0.56
	0.25	0.87	0.62	0.88	0.73
本文方法	0.25	0.81	0.54	0.85	0.64
	0.05	0.70	0.43	0.75	0.53
	0.10	0.60	0.34	0.69	0.44
	0.15	0.73	0.46	0.73	0.49

表 1 中的 2 种方法列举出 alpha 值不相同，这是因为相对比的是相同或者相近命中率下 MRV 的差距。2 种方法取到相同的命中率作对比很难，图 2 为 2 种方法的命中率和 MRV 的折线。



(a)C-81 数据集结果



(b)NASA93 数据集结果

图 2 2 种估算方法的对比

从图 2 可以看出，本文方法在图 2(a)中，命中率超过 0.7 后，MRV 的增长速度是慢于基于 MRE 的方法，在命中率低于 0.8 时，MRV 的增长速度与基于 MRE 的方法相当。本文方法在图 2(b)中，命中率超过 0.8 后，MRV 增长速度慢于基于 MRE 的方法，命中率低于 0.8 时，MRV 的增长速度与基于 MRE 的方法相当。

4.2 结果与分析

(1)相对方差逻辑

在度量不确定性度量需要一个依据，度量的准确与否是通过它所体现的区间的命中率和大小来决定，本文使用命中率和和区间相对宽度作为不确定性的依据。而区间的相对宽度本身就是本文提出的不确定性度量指标，这在逻辑上并不

矛盾。

(2)不确定性的度量优势

从实验结果中看到，本文方法在 COCOMO 模型上有一定优势，命中率高，相对宽度略窄。相对于作为对比基于直方图的方法，本文利用了 COCOMO 模型本身体现出的信息。直接对 MRE 这一类指标建模的方法做对比，一方面这一类方法由于引进新模型，不能客观地反映原来的模型在当前数据上反映出的不确定性，不具可比性。另外，这些方法实验细节没有完全透露，也没有进行如此大的重复抽取样本的实验，无法对比性能。

本文的度量方法是为度量 C81 模型在某个组织中使用时，所体现出的不确定性。这个组织的历史数据集就能体现这个组织的特点，如果新项目出现很大的变动因素，那么必定导致估算不准，不确定性的估计不足或者过大。从本文方法所取的置信度上看，命中率低于置信度，这说明一方面方差估计不足，另一方面模型在数据上的偏差影响，混合在其中，难以分离。

5 结束语

综合以上实验以及讨论，本文提出的不确定性度量指标 RV(包括基于 RV 的 MRV)能较为精确地度量出 C81 模型在某个数据集上的不确定性。本文评测方法使用区间命中率和平均相对宽度的折线图，为今后改进不确定性度量打下了一定的基础。

参考文献

[1] 李明树. 软件成本估算方法及应用[J]. 软件学报, 2007, 18(4): 775-795.

[2] 顾勋梅, 邵志清. 基于 Java 程序的功能点度量[J]. 计算机工程, 2009, 35(4): 28-30.

[3] Boehm B W. Software Engineering Economics[M]. [S. 1.]: Prentice Hall, 1981.

[4] Boehm B W. Software Cost Estimation with COCOMO II[M]. [S. 1.]: Prentice Hall, 2000.

[5] Shirabad J S, Menzies T J. The PROMISE Repository of Software Engineering Databases[EB/OL]. (2005-10-20). http://promise.site.uottawa.ca/SERepository.

[6] Jørgensen M, Sjøberg D I K. An Effort Prediction Interval Approach Based on the Empirical Distribution of Previous Estimation Accuracy[J]. Information and Software Technology, 2003, 45(3): 123-136.

[7] Jørgensen M. Regression Models of Software Development Effort Estimation Accuracy and Bias[J]. Empirical Software Engineering, 2004, 9(4): 297-314.

[8] Alessandro S S, Robert B V, Giovanni C. Using Uncertainty as a Model Selection and Comparison Criterion[C]//Proceedings of the 5th International Conference on Predictor Models in Software Engineering. Vancouver, Canada: ACM Press, 2009.

[9] Jørgensen M. Evidence-based Guidelines for Assessment of Software Development Cost Uncertainty[J]. IEEE Transactions on Software Engineering, 2005, 31(11): 942-954.

[10] Wasserman L. All of Statistics: A Concise Course in Statistical Inference[M]. [S. 1.]: Springer, 2003.

编辑 索书志

(上接第 38 页)

值。其中，PR 表示递归消除前的算法；RR 表示仅消除了 AS() 中 CASE1 处的递归，消除的方法是一门多锁法；TR 表示完全消除了递归后的算法。实验数据如表 1 所示。

表 1 3 种深度递归消除后构建占优树所消耗的时间

节点数	PR/ms	RR/ms	TR/ms
10 000	227	137	235
20 000	529	313	514
30 000	908	537	852
40 000	1 279	1 026	1 293
50 000	1 741	1 151	1 700
60 000	2 141	1 345	2 186
70 000	2 692	1 784	2 718
80 000	3 286	2 103	3 301
90 000	3 654	2 420	3 830
100 000	4 211	2 847	4 314

观察表 1 中的数据，可以发现：(1)局部的递归消除显著提升了数据结构的性能。(2)完全递归消除后，数据结构的性能和递归消除前相比，并没有明显的提升。

4.2 多函数递归消除对性能的影响

按照经验，递归消除可以提高算法的性能，且消除得越彻底，性能得提升越大。但是发现 TR 的性能与 PR 相比，没有显著差别。

分析后可以解释这种现象。RR 使用的是一门多锁法，额外开销仅是在进入迭代前后的开关锁操作，但是 TR 为了拆除多函数间的调用，模拟了系统栈的工作过程，为每个递归调用都进行了参数保存和恢复操作，所以，和 PR 具有相似的开销。但是，值得注意的是，RR 虽然有较高的效率，但是仍旧有系统栈溢出的危险。TR 却消除了这个隐患。

5 结束语

目前，多数递归消除方法都是针对单个函数的，无法解决现实中涉及多个函数的递归调用问题。本文提出了一种两步综合递归消除法，首先拆除函数间的调用，然后消除各个函数内部的递归，成功地消除了占优树的递归消除问题。在实验中，分析了在此研究中观察到得一个现象，即与完全消除递归相比，局部递归消除往往能更大程度地提高算法的性能。

参考文献

[1] 朱振元, 朱 承. 递归算法的非递归实现[J]. 小型微型计算机系统, 2003, 24(3): 567-570.

[2] 孟 林, 李 忠. 递归算法的非递归化研究[J]. 计算机科学, 2001, 28(8): 96-98.

[3] Kruse R, Tondo L, Leung B. Data Structures & Program Design in C[M]. 2nd ed. [S. l.]: Prentice Hall, 1997.

[4] Shi Chuan, Yan Zhenyu, Kevin L, et al. A Dominance Tree and Its Application in Evolutionary Multi-objective Optimization[J]. Information Sciences, 2009, 179(20): 3540-3560.

[5] 石 川, 李清勇, 史忠植. 一种快速的基于占优树的多目标进化算法[J]. 软件学报, 2007, 18(3): 505-516.

编辑 任吉慧