软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(4):933−942 [doi: 10.13328/j.cnki.jos.004964]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

软件形式化开发关键部件选取的水波优化方法*

郑宇军[1,2], 张 蓓[1], 薛锦云[2]

[1](浙江工业大学 计算机科学与技术学院,浙江 杭州 310023)

[2](江西省高性能计算重点实验室(江西师范大学),江西 南昌 330027)

通讯作者: 郑宇军, E-mail: yujun.zheng@computer.org

摘 要: 形式化方法有助于从根本上提高软件系统的质量与可靠性,但其开发成本往往过于高昂.一种折衷的办法是在软件系统中选取关键性部件进行形式化开发,但目前尚无非常有效的定量选择方法.将软件系统中的形式化开发关键部件选取建模为一个 0-1 约束规划问题,以便使用元启发式搜索方法对其进行优化求解.另外,针对该问题专门设计了一种离散水波优化(water wave optimization,简称 WWO)算法.在一个大型软件系统上的应用验证了问题模型的有效性,同时证明了 WWO 算法相对于其他若干典型元启发式搜索方法的优越性.

关键词: 形式化方法;可靠度;元启发式搜索方法;水波优化算法

中图法分类号: TP311

中文引用格式: 郑宇军,张蓓,薛锦云.软件形式化开发关键部件选取的水波优化方法.软件学报,2016,27(4):933−942. http://www.jos.org.cn/1000-9825/4964.htm

英文引用格式: Zheng YJ, Zhang B, Xue JY. Selection of key software components for formal development using water wave optimization. Ruan Jian Xue Bao/Journal of Software, 2016,27(4):933−942 (in Chinese). http://www.jos.org.cn/1000-9825/4964.htm

Selection of Key Software Components for Formal Development Using Water Wave Optimization

ZHENG Yu-Jun[1,2], ZHANG Bei[1], XUE Jin-Yun[2]

[1](College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

[2](Jiangxi Provincial Key Laboratory of High Performance Computing (Jiangxi Normal University), Nanchang 330027, China)

Abstract: Formal methods contribute to the fundamental improvement of software quality and reliability, but this methodology is often very expensive. A compromise is to select and apply formal methods to only a subset of key components of the software system. However, currently there are few effective approaches for such selection process. This paper proposes a 0-1 constrained programming model for selecting key components for formal development, which enables the use of metaheuristic search methods to effectively solve the selection problem. The paper also designs a discrete water wave optimization (WWO) algorithm for the problem. The application to a large-scale software system validates the effectiveness of the proposed problem model, and demonstrates that the WWO algorithm outperforms some other typical metaheuristic search methods.

Key words: formal method; reliability; metaheuristic search method; water wave optimization (WWO)

软件形式化方法是指建立在严格数学模型上、具有精确数学语义的软件系统开发方法,它能够从根本上提高软件系统的质量与可靠性.受“软件危机”以及硬件领域形式化方法发展的影响,软件形式化方法自 20 世纪 80 年代末以来得到了学术界和工业界的广泛关注[1].但是,关于何时、何处以及何种程度上在软件系统开发中引入

* 基金项目: 国家自然科学基金(61020106009, 61105073, 61272075, 61473263)
Foundation item: National Natural Science Foundation of China (61020106009, 61105073, 61272075, 61473263)
收稿时间: 2015-06-24; 修改时间: 2015-10-15; 采用时间: 2015-11-20; jos 在线出版时间: 2016-01-13
CNKI 网络优先出版: 2016-01-14 13:16:29, http://www.cnki.net/kcms/detail/11.2560.TP.20160114.1316.012.html

形式化方法,仍然存在较大的争论.很多软件开发机构对于使用形式化方法仍然持谨慎甚至怀疑的态度,其原因在于软件形式化方法自身在理论和实践上的一些局限性[2],其中最主要的一点就是软件形式化方法往往需要付出相对高昂的代价[3],这使得对大型软件系统进行全面形式化开发通常是不现实的.

一种折衷的办法是,在软件系统开发中部分引入形式化方法.Easterbrook和Callahan[4]实践了对大规模安全关键性系统的部分规约进行形式化验证,但未介绍应如何选择这些部分规约,实践时完全依靠主观经验.Russo[5]提出了一些指导性原则用于选取重要软件部件进行形式化开发,主要包括安全关键性部件、具有复杂控制逻辑的部件、手工开发容易出错的部件,但这些原则大都是定性的,缺乏量化的选择依据.薛锦云提出的形式化方法PAR则强调对软件中的复杂算法进行形式化开发,特别是对算法循环不变式进行推导和验证[6,7].在文献[8]中,我们首次提出了一种基于软件度量的选择方法,即:根据功能点数量、耦合度、继承复杂度等度量给定一个阈值,然后选取超过阈值的部件进行形式化开发.这虽然是一种定量的方法,但由于软件度量本身的局限性[9],选取的效果与实际期望之间往往还存在较大的距离.

基于搜索的软件工程[10]是研究利用元启发搜索方法来求解软件工程中的优化问题,相关技术已被应用在软件工程的多个领域[11],包括软件需求工程、费用估算、编译器优化、代码优化,特别是软件测试[12].但到目前为止,元启发搜索方法在形式化软件工程相关问题中的研究还极为罕见.

针对大型软件系统中形式化开发的关键部件选取(selection of key software components,简称SKSC)问题,本文提出了一个非线性0-1规划问题模型,其目标是在给定的开发工作量和费用约束下最大化整个系统的(预期)可靠性水平.为了有效地求解该问题,本文提出了一种离散水波优化(water wave optimization,简称WWO)算法[13],其中设计了特定的传播、折射和碎浪操作.在一个大型软件系统上的实验结果表明,所设计的离散WWO算法优于其他若干流行的元启发式算法.

1 形式化开发关键部件选取的优化问题模型

1.1 决策变量

本文考虑对选取的软件关键部件进行完整的形式化,即,从形式化的需求规格说明、验证直至代码生成(精化/转换)[14−16].形式化开发的最小部件单位为数据类型(以下简称为类).给定一个软件系统的总体结构,我们首先筛选出3种类:一是无需专门开发的类(如类库中已有的类),二是必须进行形式化开发的类(如执行安全攸关功能的类),三是不适合或尚无法进行形式化开发的类(如富媒体界面类).记剩余的类的集合为C,集合大小为n,需要从中选取用于形式化开发的类的子集.该问题是一个0-1规划问题,即:问题的解可编码为一个长度为n的向量,向量第i维为1表示部件i进行形式化开发,为0表示非形式化开发.

1.2 目标函数

对于已有的类,通过统计分析估算其可靠度:对于使用形式化方法开发的类,其可靠度直接估算为1;对于不使用形式化方法开发的类,可分别基于代码行数或功能点来估算其可靠度[17].不过我们发现,将二者组合起来的估算结果更为准确.对于类x,记其预计(需要手工编写的)代码行数为$L(x)$,分配的功能点集合为$F(x)$,则x的可靠度按如下经验公式进行估算:

$$R(x)=1-\gamma\left[0.2\left(\frac{L(x)}{L_{\max}}\right)^{a}+0.8\frac{\sum_{f\in F(x)}b^{O(f)}}{b^{5}\mid F(x)\mid}\right] \tag{1}$$

其中,$L_{\max}$为一个独立类的代码行数建议上限,$O(f)$为对功能点f的复杂度评值(在1~5之间,1为最简单,5为最复杂),a和b是两个常数(本文中分别取经验值0.72和1.18),γ是代表开发团队能力的一个系数(在0~1之间,值越小代表能力越强).

基于部件可靠度来估算整个系统可靠度的方法大致可以分为3类.

(1) 基于结构复合的方法[18−20],其优点是计算方式简单,但缺点是假定每个部件的内部状态是固定的,因

此估算准确度相对较低;

(2) 基于概率统计的方法,如基于状态转移概率和故障传播概率的方法[21−23],其优点是估算准确度相对较高,但缺点是计算量相对较大,且往往需要通过测试来统计先验概率值,不适合于尚未开发的软件;

(3) 基于软计算的方法,如基于神经网络和模糊计算的方法[24−26],其优点是几乎不需要关于系统的假定条件,但缺点是计算量较大,而且一般需要大量历史数据来进行训练.

后两类方法的前提条件在本文的场景中难以满足,且软件可靠度本身无法做到精确的估算,而进化算法的大量迭代也能部分补偿适应度函数的不准确度[27],故这里采用第 1 类方法,复合时考虑以下 3 种情况.

(1) K 个部件 $x_1,x_2,\ldots,x_K$ 的顺序(串联)组合(如图 1 所示),其复合结构 x 的可靠度为

$$R(x)=\prod_{k=1}^{K}R(x_k) \tag{2}$$

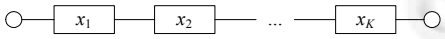


Fig.1 Serial composition of components

图 1 部件的串联结构

(2) K 个部件 $x_1,x_2,\ldots,x_K$ 的并行(并联)组合(如图 2 所示),其复合结构 x 的可靠度为

$$R(x)=1-\prod_{k=1}^{K}(1-R(x_k)) \tag{3}$$

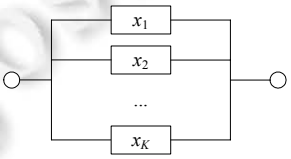


Fig.2 Parallel composition of components

图 2 部件的并联结构

(3) 一个部件 x_0 对 K 个部件 $x_1,x_2,\ldots,x_K$ 的选择调用组合(如图 3 所示),设其中每个部件的调用概率为 h_k ($1\leqslant k\leqslant K$),其复合结构 x 的可靠度为

$$R(x)=R(x_0)\sum_{k=1}^{K}h_kR(x_k) \tag{4}$$

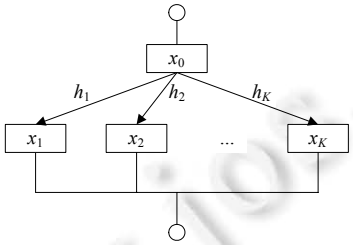


Fig.3 Conditional composition of components

图 3 部件的选择复合结构

其他更为复杂的结构大都可以由以上 3 种结构复合而成.通过自底向上地逐级复合,最后估算出整个系统 $S(\boldsymbol{X})$的可靠度 $R(S(\boldsymbol{X}))$[18−20],其中,$\boldsymbol{X}$ 为问题的决策向量.问题的目标就是最大化整个系统的可靠度.

$$\mathrm{Max}R(S(\boldsymbol{X})) \tag{5}$$

对于大规模的软件系统,该目标函数涉及到大量部件及其复合结构的可靠度计算,是一个高度复杂、非线性的目标函数.

1.3 约束条件

考虑开发时间和费用两方面的约束条件,二者都是基于开发工作量来进行估算的.软件成本及工作量估算有算法模型、类比、回归分析、神经网络等多种方法[28].原则上,只要在估算时能够区分形式化和非形式化开发的区别,任何一种方法都可以用于本文的问题模型.给定问题的一个决策向量 $\boldsymbol{X}$,记其中形式化开发的类的子集(即 X_i=1 的元素集合)为 $C_F(\boldsymbol{X})$,工作量估算函数为 p_1;非形式化开发的类的集合(即 X_i=0 的元素集合)为 $C_N(\boldsymbol{X})$,工作量估算函数为 p_2,则 $C_F(\boldsymbol{X})$和 $C_N(\boldsymbol{X})$的开发工作量可分别计算为

$$p_F(\boldsymbol{X})=\sum_{x_i\in C_F(\boldsymbol{X})}p_1(x_i) \tag{6}$$

$$p_N(\boldsymbol{X})=\sum_{x_i\in C_N(\boldsymbol{X})}p_2(x_i) \tag{7}$$

我们的研究主要采用 COCOMO-II 模型[29]及其在形式化开发中的扩展模型[30]来估算开发工作量,单位为人·月(PM).不论是 p_1 还是 p_2,都是基于如下的基础模型来估算每个类 x 的工作量 $p(x)$:

$$p(x)=A\cdot\left(1+\frac{V(x)}{100}L(x)\right)^B\cdot E \tag{8}$$

其中,A 为校准因子,B 为工作量比例因子,E 为成本因子,$V(X)$为类 x 的需求演化和变更因子.形式化开发与非形式化开发工作量的区别,主要就在于前 3 个因子取值的不同.

设 $C_F(\boldsymbol{X})\cup C_N(\boldsymbol{X})$中所有类的开发工作量上限为 $P_{\max}$,则对工作量的约束可表达为

$$p_F(\boldsymbol{X})+p_N(\boldsymbol{X})\leqslant P_{\max} \tag{9}$$

再设非形式化开发中每 PM 的平均费用为 c_1,形式化开发的平均费用为 c_2,所有类的开发费用上限为 $C_{\max}$,则对费用的约束可表达为

$$c_1p_F(\boldsymbol{X})+c_2p_N(\boldsymbol{X})\leqslant C_{\max} \tag{10}$$

值得注意的是:上述对开发工作量和费用的估算都是针对基础类进行的,而不包括各级部件复合乃至最终系统集成的费用,因为我们只区分基础类的形式化和非形式化开发,而不同的选择策略对后期的复合和集成费用不会产生区别.因此,上述估算策略并不适用于对整个软件系统的开发工作量和费用的精确估算.

2 求解问题的水波优化算法

前一小节提出的 SKSC 问题是一个复杂的非线性 0-1 约束规划问题,可应用各种元启发式方法进行求解.本文设计了一种基于 WWO 的求解算法.

2.1 基本水波优化算法

WWO 是一种基于浅水波理论[31]的启发式算法,它将问题的搜索空间类比为海床,将问题的每个解类比为一个“水波”,水波的适应度与其到海床的垂直距离成反比:距离海平面越近的点,对应的解越优,相应的水波能量越高,那么水波的波高 h 更大、波长λ更小,如图 4 所示[13].

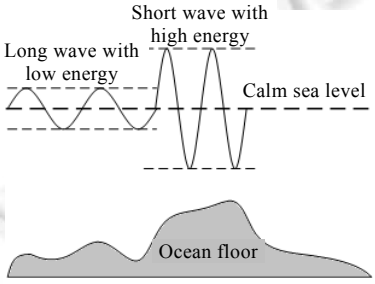


Fig.4 Illustration of the WWO algorithm model

图 4 水波优化模型示意图

这使得较优的解在较小的范围内进行搜索,而较差的解在较大的范围内进行搜索,从而促进整个种群不断向更优的目标进化.

WWO 算法在初始化时将每个水波的 h 值设置为一个整数常量 $h_{\max}$,λ值设置为 0.5.在进化过程中,WWO 提供了 3 种具体操作,即传播、折射和碎浪.

(1) 传播

在算法每次迭代过程中,种群中的每个水波都传播一次.设问题的维数为 n,当前水波 $\boldsymbol{X}$ 在每一维的位置记为 $X(d)$,其传播后在每一维的新位置为

$$X'(d)=X(d)+rand(-1,1)\cdot\lambda L(d) \tag{11}$$

其中,$rand(-1,1)$代表[−1,1]范围里的一个均匀分布随机数,$L(d)$代表搜索空间在第 d 维的长度($1\leqslant d\leqslant n$).如果某一维的新位置超出了有效范围,则将其随机设置为有效范围内的一个位置.

令 f 表示问题的适应度函数,传播后计算新波 $\boldsymbol{X}'$的适应度值,如果 $f(\boldsymbol{X}')>f(\boldsymbol{X})$,则 $\boldsymbol{X}'$在种群中取代 $\boldsymbol{X}$,其波高重置为 $h_{\max}$;反之,$\boldsymbol{X}$ 会被保留,且其波高 h 由于能量的损耗而减 1.

每次迭代后,算法按下式对种群中的每个水波 X 的波长进行更新.

$$\lambda=\lambda\cdot\alpha^{-(f(X)-f_{\min}+\varepsilon)/(f_{\max}-f_{\min}+\varepsilon)} \tag{12}$$

其中,$f_{\max}$ 和 $f_{\min}$ 分别表示当前种群中的最大和最小适应度值,参数α表示波长的衰减系数,ε是一个极小的正数(以避免分母为 0 的情况发生).

(2) 折射

当某个水波 X 的 h 值递减为 0 时,对其折射操作以避免搜索停滞,折射后每一维的位置计算公式如下:

$$X'(d)=N\left(\frac{X^*(d)+X(d)}{2},\frac{|X^*(d)-X(d)|}{2}\right) \tag{13}$$

其中,$\boldsymbol{X}^*$表示目前位置所找到的最优解,$N(\mu,\sigma)$表示均值为μ、方差为σ的高斯随机数.折射后新波 $\boldsymbol{X}'$的波高同样重置为 $h_{\max}$,波长则按下式进行更新,这也会使得解的适应度与波长成反比.

$$\lambda'=\lambda\frac{f(\boldsymbol{X})}{f(\boldsymbol{X}')} \tag{14}$$

(3) 碎浪

水波能量的不断增加会使其波峰变得越来越陡峭,直至破碎成一连串的孤立波.WWO 算法对每个新找到的最优解 $\boldsymbol{X}^*$执行碎浪操作,具体方式是:先随机选择 k 维(这里,k 是介于 1 和一个预定义参数 $k_{\max}$ 之间的一个随机数),在每一维 d 上产生一个孤立波 X'.

$$X'(d)=X(d)+N(0,1)\cdot\beta L(d) \tag{15}$$

其中,参数β表示碎浪系数.如果生成的所有孤立波的适应度值均不优于 $\boldsymbol{X}^*$,则保留 $\boldsymbol{X}^*$;否则,将 $\boldsymbol{X}^*$替换为最优的一个孤立波.

基本 WWO 算法步骤描述如下:

第 1 步. 随机初始化一个种群,计算其中每个解 x 的适应度值 $f(x)$,找出其中的最优解 x^*;

第 2 步. 若终止条件满足,算法结束,返回 x^*;

第 3 步. 对种群中的每个解 x 执行如下过程:

第 3.1 步. 对 x 执行传播操作,得到一个新波 x';

第 3.2 步. 若 $f(x')>f(x)$,则用 x'替换 x:

第 3.2.1 步. 若 $f(x')>f(x^*)$,则用 x'替换 x^*,并对 x 执行碎浪操作;

第 3.3 步. 否则,将 x 的波高 h 值减 1;

第 3.3.1 步. 若 $h=0$,对 x 执行折射操作;

第 3.4 步. 更新种群中所有解的波长,而后转第 2 步;

2.2 离散水波优化算法

基本 WWO 算法适用于连续空间中的全局优化问题.SKSC 问题属于组合优化问题,因此需要对 WWO 的算子进行调整.

(1) 传播

WWO 传播操作的原则就是使适应度值高的个体具有短波长,从而在较小的范围进行搜索;反之,适应度值低的个体在较大范围内进行搜索.针对 SKSC 问题,我们基于邻域结构来定义传播操作.给定任意一个解向量 $\boldsymbol{X}$,其直接邻域解可通过两种方式得到:一是将一个非形式化开发的类改为形式化开发(对应元素由 0 变为 1),二是将一个形式化开发的类改为非形式化开发(对应元素由 1 变为 0).那么对于维数为 n 的问题,邻域结构的大小就是 n.算法初始化时,每个解的波长设为 $n^{1/4}$.对于波长为λ的解 $\boldsymbol{X}$,其传播操作就是随机生成它的一个λ步邻域解.每次迭代后,每个解的波长按下式进行更新.

$$\lambda = n^{0.5(f_{\max}-f(X)+\varepsilon)/(f_{\max}-f_{\min}+\varepsilon)} \tag{16}$$

特别地,当 $f(\boldsymbol{X})$取最大值 $f_{\max}$ 时,λ=1,传播操作生成一个一步(直接)邻域解;当 $f(\boldsymbol{X})$取最小值 $f_{\min}$ 时,$\lambda=n^{1/2}$,传播操作随机执行$|n^{1/2}|$步 0-1 变换.

(2) 折射

WWO 折射操作的本质是向当前最优解 $\boldsymbol{X}^*$学习.针对 SKSC 问题,我们定义在给定解 $\boldsymbol{X}$ 上的传播操作执行方式如下:

i) 计算 $C_1=C_F(\boldsymbol{X}^*)\backslash C_F(\boldsymbol{X}),C_2=C_N(\boldsymbol{X}^*)\backslash C_N(\boldsymbol{X})$;

ii) 计算 $K=N(m,m-\min(|C_1|,|C_2|))$,其中,$m=(|C_1|+|C_2|)/2$;

iii) 在 $C_F(\boldsymbol{X})\backslash C_F(\boldsymbol{X}^*)$中随机取 $\min(K,|C_F(\boldsymbol{X})\backslash C_F(\boldsymbol{X}^*)|)$个元素,将其值由 1 变为 0;

iv) 在 $C_N(\boldsymbol{X})\backslash C_N(\boldsymbol{X}^*)$中随机取 $\min(K,|C_N(\boldsymbol{X})\backslash C_N(\boldsymbol{X}^*)|)$个元素,将其值由 0 变为 1.

也就是说,传播操作是将 $\boldsymbol{X}$ 中部分与 $\boldsymbol{X}^*$不同的解分量改为与 $\boldsymbol{X}^*$相同,且由 1 变为 0 的分量数与由 0 变为 1 的分量数大致持平.

(3) 碎浪

针对 SKSC 问题上定义的碎浪操作也是基于邻域结构的:每当传播操作得到一个新的当前最优解 $\boldsymbol{X}^*$时,生成 $\boldsymbol{X}^*$的 k 个一步和二步邻域解;如果其中包含比 $\boldsymbol{X}^*$更优的解,则再次更新当前最优解.

(4) 约束处理

这里,对 SKSC 问题的约束条件(9)和约束条件(10)采用惩罚函数的处理方法.对于每个解 $\boldsymbol{X}$,计算其对约束(9)和约束(10)的违反度 $\Psi_1(\boldsymbol{X})=\max(p_F(\boldsymbol{X})+p_N(\boldsymbol{X})-P_{\max},0)$, $\Psi_2(\boldsymbol{X})=\max(c_1p_F(\boldsymbol{X})+c_2p_N(\boldsymbol{X})-C_{\max},0)$,再按下式计算其适应度值:

$$f(\boldsymbol{X})=R(S(\boldsymbol{X}))-M_1\Psi_1(\boldsymbol{X})-M_2\Psi_2(\boldsymbol{X}) \tag{17}$$

其中,M_1 和 M_2 是两个很大的正数,其值通常要比 $P_{\max}$ 和 $C_{\max}$ 高几个数量级.

(5) 复杂度分析

设目标函数 f 的计算复杂度为 $O(f)$,邻域变换和随机数生成的计算复杂度均用常数 c 表示,算法种群规模为 N.每次传播操作执行λ步邻域变换并对得到的邻域解进行估值,故每次迭代中传播操作的时间复杂度最坏情况下不超过 $O(N(nc+O(f)))$.折射操作产生一个介于当前波高为 0 的个体与当前最优个体之间的新个体,每次操作的时间复杂度为 $O(c+O(f))$.碎浪操作只在当前最优个体上执行,每次操作生成其 k 个邻域解,时间复杂度不超过 $O(k_{\max}(c+O(f)))$.每次迭代时,折射操作和碎浪操作不可能同时作用于同一个个体,故算法每次迭代的时间复杂度最坏情况下不超过 $O(N(nc+k_{\max}c+k_{\max}O(f)))$.

若设置算法最大迭代次数为 T,则算法的计算复杂度为 $O(TN(nc+k_{\max}c+k_{\max}O(f)))$.

(6) 操作示例

假定问题维度为 6,对一个编码为 101011 的解 $\boldsymbol{X}$,设波长为 2,则传播操作执行二步随机邻域变换,不妨设先在第二维、后在第五维上变换,则变换后的结果为 111001.若该解优于 $\boldsymbol{X}$,则用其取代 $\boldsymbol{X}$.

再设当前最优解 $\boldsymbol{X}^*$为 011010,则 $\boldsymbol{X}$ 与 $\boldsymbol{X}^*$在第 1 位、第 5 位、第 6 位上不同,则折射操作在 $\boldsymbol{X}$ 的第 1 位和第 6 位中随机选取一个(不妨设选取第 6 位)由 1 变 0,并将 $\boldsymbol{X}$ 的第 5 位由 0 变 1,故折射后的结果为 111010.

对 $\boldsymbol{X}^*$执行碎浪操作,设 $k=rand(1,k_{\max})=3$,则随机生成其 3 个一步或二步邻域解(如 011110,011100,011000);若其中包含比 $\boldsymbol{X}^*$更优的解,则用其替换 $\boldsymbol{X}^*$.

3 计算实验

上述模型和算法应用于一个大型军事指挥控制软件的开发决策,其特点是可靠性要求较高、开发时间又较为紧迫.经初步设计,确定该系统包含 1 364 个基本类,其中 1 168 个类尚不确定是否要使用形式化方法来进行开发.该形式化开发部件的选取问题建模为第 1 节中的 SKSC 问题,并使用第 2 节中的离散 WWO 算法来进行求解.为验证算法性能,还在该问题实例上运行以下 4 种流行算法来进行比较.

- 一种用于子集选取的遗传算法(genetic algorithm,简称 GA)[32];
- 二进制粒子群优化(binary particle swarm optimization,简称 BPSO)算法[33];
- 一种改进的二进制 PSO 算法 IBPSO,它将 PSO 搜索与 Lambda 迭代相结合[34];
- 二进制差分搜索(binary differential evolution,简称 BDE)算法[35].

实验环境为一台使用 Intel Core i5-2430M 处理器、4GB DDR3 内存、Windows 7 操作系统的计算机,算法程序均使用 Visual Studio 2010 环境开发.所有算法的种群大小均设为 50,WWO 设置为 $h=6,k_{\max}=6$,其他 4 种算法的参数均按其原始文献进行设置.为公平比较,对所有算法设定统一的终止条件,即,适应度估值次数达到 50 000 次.每种算法随机运行 30 次,统计其各次求得最优解(最大可靠度,以%为单位)的中位数(median)、平均值(mean)、最大值(max)、最小值(min)、方差(std)以及平均运行时间(time,以秒为单位),结果见表 1.其他 4 种算法的结果与 WWO 的结果还分别进行了秩和校验,均值前的上标“+”表示 WWO 算法的结果显著优于该算法(置信度为 95%).

Table 1 Computational results of the five algorithms on the given SKSC problem instance

表 1 5 种算法在给定软件系统形式化开发部件选取问题上的运行结果

	GA	BPSO	IBPSO	BDE	WWO
Median	$^{+}$93.88	$^{+}$92.89	$^{+}$94.36	95.32	**95.63**
Mean	94.27	93.13	94.16	**95.12**	94.85
Max	95.22	94.87	95.50	95.73	**95.98**
Min	91.85	91.26	91.67	93.69	**93.73**
Std	1.71	2.21	2.54	1.26	1.29
Time	3 450	3 285	3 372	3 107	3 163

从计算时间来看,GA 的运行时间最长,BDE 运行时间最短,WWO 的运行时间略长于 BDE 但短于其余 3 种算法.这说明 WWO 的算法速度也是较为理想的.总的来说,5 种算法的运行时间相差不大,这是因为问题的适应度计算较为复杂,占用了算法的绝大多数运行时间,而 5 种算法的适应度估值次数上限是相同的.而且从算法每次接近 1 小时的运行时间来看,该问题的求解难度也是相当大的.

从计算结果来看:可靠度,median,max 和 min 值最高的都是 WWO 算法,而 mean 值最高的是 BDE 算法(这主要是因为 WWO 的结果中出现了一个显著噪点而拉低了 mean 值).BPSO 算法的综合表现最差,IBPSO 对其进行了一定改进,但求解效果仍不如 BDE 和 WWO;GA 的性能大致介于 BPSO 和 IBPSO 之间.从统计检验的结果来看,WWO 算法的结果比 GA,BPSO 和 IBPSO 都有显著提升,仅与 BDE 之间无显著性差异.按 30 次运行结果的中位数来比较,WWO 求得的可靠度比 GA,BPSO,IBPSO,BDE 分别高 1.86%,2.95%,1.35%,0.33%;而按 30 次运行结果的最大值来比较,WWO 求得解的可靠度仍比其余 4 种算法分别高 0.80%,1.17%,0.50%,0.26%.对于绝大多数重要软件系统,这种程度的可靠性提升对系统拥有者来说是极为重要的,在很多情况下,甚至能够决定系统关键任务的成败[36].

从理论上分析,SKSC 问题不同解的目标函数值相差可能很大,解空间易形成多峰分布,而且峰附近可能存在较大起伏.上述算法都是将一个 n 维 0-1 规划问题的解编码为一个长度为 n 的二进制串.BPSO 仍采用传统粒

子群中向个体最优和全局最优的学习公式,而后将公式产生的实数位舍入为二进制位,这种转换策略在处理长串时学习效果很不理想.IBPSO和BDE对此进行了改进,增强了全局搜索能力,但在起伏较大的峰附近的局部搜索能力较弱.GA 中的操作最适合本问题的编码方式,但它与其他很多 GA 算法一样,都容易过早地陷入局部最优.而 WWO 算法能够基于波长机制很好地平衡全局探索和局部开发,更容易通过水波群来定位多峰,通过折射操作避免搜索停滞,并通过碎浪操作来增强在起伏较大的峰附近的局部搜索,因此在 SKSC 问题上表现出了极佳的性能.

此外,我们还采用了文献[8]中基于软件度量的方法来对系统中的形式化开发部件进行选取,并基于选取结果来进行系统可靠度估算,得出的结果是91.05%,该值低于上述所有元启发式搜索算法的最差结果.这充分说明了本文所提出的 SKSC 优化问题模型的有效性以及所设计的算法的优越性.相比于基于软件度量的定量方法,WWO 求得最佳结果的可靠度提升了 5.41%,这是一个相当正面的结果.

该系统的开发决策最终采纳了WWO求得的最佳解(估算可靠度为95.98%).基于9个月的实际运行数据分析,系统的实测可靠度为94.80%.这种偏差的主要原因在于模型参数经验取值的误差,此外也存在系统用户早期使用的磨合问题.但即便如此,预计可靠度与实测可靠度的误差仍不超过 1.25%,这也进一步说明了问题模型的有效性.总的说来,系统拥有者对基于本文问题模型和算法所做出的形式化开发决策非常满意.

4 结束语

形式化方法有助于从根本上提高软件系统的质量与可靠性,但其成本往往远高于传统的开发方法.本文提出了一个在软件系统中选取关键部件进行形式化开发的SKSC优化问题模型,并设计了一种离散WWO算法来对其进行有效求解.通过在一个典型系统上的应用,证明了本文提出的模型和算法的有效性.

本文针对 SKSC 智能优化进行了初步研究,目前正在以下几个方面开展进一步的工作.

- 当前模型考虑的是对选取部件进行完整的形式化开发,对此可作进一步细化,考虑不同程度的形式化方法的应用,如形式化需求导出、形式化规格说明、形式化验证、形式化代码推导/转换等,都可单独地应用到不同的部件上;
- 当前模型只对系统整体的可靠性进行一个粗略估算,下一步可考虑系统不同部分以及执行不同任务的不同可靠性需求,从而为形式化开发部件的选取提供更为精准的指导;
- 当前模型对开发成本的估算也较为粗略,可融合神经网络、模糊推理等技术进行更精确的估算.

此外,当前模型和算法只在一个特定软件系统上进行了测试.我们正在将其扩展应用于更多数量和类型的软件系统,以便为模型和算法改进提供更充足的数据支持.

References:

[1] Vienneau RL. A review of formal methods. In: Dorfman M, Thayer RH, eds. Proc. of the Software Engineering. Washington: IEEE Computer Society Press, 1996. 181−192.

[2] Kneuper R. Limits of formal methods. Formal Aspects of Computing, 1997,9(4):379−394. [doi: 10.1007/BF01211297]

[3] Woodcock J, Larsen PG, Bicarregui J, Fitzgerald J. Formal methods: Practice and experience. ACM Computing Surveys (CSUR), 2009,41(4):1−40. [doi: 10.1145/1592434.1592436]

[4] Easterbrook S, Callahan J. Formal methods for verification and validation of partial specifications: A case study. Journal of Systems and Software, 1998,40(3):199−210. [doi: 10.1016/S0164-1212(97)00167-2]

[5] Russo AG. Partial applications of formal methods. In: Boulanger JL, ed. Proc. of the Industrial Use of Formal Methods: Formal Verification. Hoboken: John Wiley & Sons, 2013. 195−214. [doi: 10.1002/9781118561829.ch6]

[6] Xue JY. Two new strategies for developing loop invariants and their applications. Journal of Computer Science and Technology, 1993,8(2):147−154. [doi: 10.1007/BF02939477]

[7] Xue JY. A unified approach for developing efficient algorithmic programs. Journal of Computer Science & Technology, 1997,12(4): 314−329. [doi: 10.1007/BF02943151]

[8] Zheng YJ, Wang JQ, Wang K, Xue JY. Partially introducing formal methods into object-oriented development: Case studies using a metrics-driven approach. In: Misra J, Nipkow T, Sekerinski E, eds. Proc. of the Formal Methods. Berlin: Springer-Verlag, 2006. 190−204. [doi: 10.1007/11813040_14]

[9] Kan SH. Metrics and Models in Software Quality Engineering. 2nd ed., Reading MA: Addison-Wesley, 2003.

[10] Harman M, Jones BF. Search-Based software engineering. Information and Software Technology, 2001,43(14):833−839. [doi: 10.1016/S0950-5849(01)00189-6]

[11] Harman M. The current state and future of search based software engineering. In: Briand LC, Wolf AL, eds. Proc. of the Future of Software Engineering. Washington: IEEE Computer Society, 2007. 342−357. [doi: 10.1109/FOSE.2007.29]

[12] McMinn P. Search-Based software test data generation: A survey. Software Testing, Verification and Reliability, 2004,14(2): 105−156. [doi: 10.1002/stvr.294]

[13] Zheng YJ. Water wave optimization: A new nature-inspired metaheuristic. Computers & Operations Research, 2015,55:1−11. [doi: 10.1016/j.cor.2014.10.008]

[14] Xue JY. A unified approach for developing efficient algorithmic programs. Journal of computer Science and Technology, 1997, 12(4):314−329. [doi: 10.1007/BF02943151]

[15] Sekerinski E, Kaisa S, eds. Program Development by Refinement: Case Studies Using the B Method. Berlin: Springer-Verlag, 1998.

[16] Zheng YJ, Ling HF, Xue JY. Combinatorial optimization problem reduction and algorithm derivation. Ruan Jian Xue Bao/Journal of Software, 2011,22(9):1985−1993 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3948.htm [doi: 10.3724/SP.J.1001.2011.03948]

[17] Musa JD, Iannino A, Okumoto K. Software Reliability: Measurement, Prediction, Application. New York: McGraw-Hill, 1987.

[18] Zhu H. Toward a relationship between software reliability estimation and complexity analysis. Ruan Jian Xue Bao/Journal of Software, 1998,9(9):713−717 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/9/713.htm

[19] Hamlet D, Mason D, Woit D. Theory of software reliability based on components. In: Müller HA, ed. Proc. of the 23rd Int'l Conf. on Software Engineering. Washington: IEEE Computer Society, 2001. 361−370. [doi: 10.1109/ICSE.2001.919109]

[20] Xu SY. Design and Analysis of Dependable Computational Systems. Beijing: Tsinghua University Press, 2006 (in Chinese).

[21] Smidts C, Stutzke M, Stoddard RW. Software reliability modeling: An approach to early reliability prediction. IEEE Trans. on Reliability, 1998,47(3):268−278. [doi: 10.1109/24.740500]

[22] Wang WL, Pan D, Chen MH. Architecture-based software reliability modeling. Journal of Systems and Software, 2006,79(1): 132−146. [doi: 10.1016/j.jss.2005.09.004]

[23] Palviainen M, Evesti A, Ovaska E. The reliability estimation, prediction and measuring of component-based software. Journal of Systems and Software, 2011,84(6):1054−1070. [doi: 10.1016/j.jss.2011.01.048]

[24] Kiran NR, Ravi V. Software reliability prediction by soft computing techniques. Journal of Systems and Software, 2008,81(4): 576−583. [doi: 10.1016/j.jss.2007.05.005]

[25] Guo P. Computational Intelligence in Software Reliability Engineering. Beijing: Science Press, 2012 (in Chinese).

[26] Bhuyan MK, Mohapatra DP, Sethi S. Measures for predicting software reliability using time recurrent neural networks with back-propagation. ACM SIGSOFT Software Engineering Notes, 2015,40(5):1−8. [doi: 10.1145/2815021.2815030]

[27] Mansanne F, Carrere F, Ehinger A, Schoenauer M. Evolutionary algorithms as fitness function debuggers. In: Raś ZW, Skowron A, eds. Proc. of the Foundations of Intelligent Systems. Berlin: Springer-Verlag, 1999. 639−647. [doi: 10.1007/BFb0095153]

[28] Li MS, He M, Yang D, Su DF, Wang Q. Software cost estimation method and application. Ruan Jian Xue Bao/Journal of Software, 2007,18(4):775−795 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/775.htm

[29] Boehm BW, Abts C, Brown AW, Chulani S, Clark BK, Horowitz E, Madachy R, Reifer D, Steece B. Software Cost Estimation with COCOMO II. New York: Prentice Hall, 2000.

[30] Zheng YJ, Wang K, Xue JY. An extension of COCOMO II for the B-method. In: Sullivan K, Kazman R, eds. Proc. of the 7th Int'l Workshop on Economics Driven Software Engineering Research. New York: ACM Press, 2006. 11−14. [doi: 10.1145/1139113.1139118]

[31] Huang H. Dynamics of Surface Waves in Coastal Waters: Wave-Current-Bottom Interactions. New York: Springer-Verlag, 2009.

[32] Tan F, Fu X, Zhang Y, Bourgeois AG. A genetic algorithm-based method for feature subset selection. Soft Computing, 2008,12(2): 111−120. [doi: 10.1007/s00500-007-0193-8]

[33] Kennedy J, Eberhart RC. A discrete binary version of the particle swarm algorithm. In: Tien JM, ed. Proc. of the IEEE Int'l Conf. on Systems, Man, and Cybernetics. Piscataway: IEEE, 1997. 4104−4108. [doi: 10.1109/ICSMC.1997.637339]

[34] Yuan X, Nie H, Su A, Wang L, Yuan Y. An improved binary particle swarm optimization for unit commitment problem. Expert Systems with Applications, 2009,36(4):8049−8055. [doi: 10.1016/j.eswa.2008.10.047]

[35] Pampara G, Engelbrecht AP, Franken N. Binary differential evolution. In: Proc. of the IEEE Congress on Evolutionary Computation. Piscataway: IEEE, 2006. 1873−1879. [doi: 10.1109/CEC.2006.1688535]

[36] Schneidewind NF. Reliability modeling for safety-critical software. IEEE Trans. on Reliability, 1997,46(1):88−98. [doi: 10.1109/24.589933]

附中文参考文献:

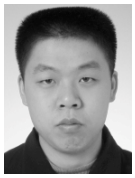
[16] 郑宇军,薛锦云,凌海风.组合优化问题简约与算法推演.软件学报,2011,22(9):1985−1993. http://www.jos.org.cn/1000-9825/3948.htm [doi: 10.3724/SP.J.1001.2011.03948]

[18] 朱鸿.软件可靠性估计与计算复杂性的关系浅析.软件学报,1998,9(9):713−717. http://www.jos.org.cn/1000-9825/9/713.htm

[20] 徐拾义.可信计算系统设计与分析.北京:清华大学出版社,2006.

[25] 郭平.软件可靠性工程中的计算智能方法.北京:科学出版社,2012.

[28] 李明树,何梅,杨达,舒风笛,王青.软件成本估算方法及应用.软件学报,2007,18(4):775−795. http://www.jos.org.cn/1000-9825/18/775.htm



郑宇军(1979－),男,福建莆田人,博士,副教授,CCF 会员,博士生导师,主要研究领域为软件形式化方法,智能优化算法.

张蓓(1990－),女,硕士生,CCF 学生会员,主要研究领域为智能优化算法.

薛锦云(1947－),男,教授,博士生导师,CCF 高级会员,主要研究领域为软件形式化与自动化.