一种基于 Kalman 滤波和粒子群优化的测试数据生成方法

薛 猛[1],姜淑娟[1,2],张争光[1],钱俊彦[2],张艳梅[1,3],曹鹤玲[4]
(1. 中国矿业大学计算机科学与技术学院,江苏徐州 221116;2. 桂林电子科技大学广西可信软件重点实验室,广西桂林 541004;
3. 南京大学计算机软件新技术国家重点实验室,江苏南京 210093;4. 河南工业大学信息科学与工程学院,河南郑州 450001)

摘 要: 为减少进化代数,提高路径覆盖成功率,提出了多邻域 Kalman 滤波 PSO 测试数据生成方法. 在该方法中将粒子固定划分到不同邻域中,各邻域内指定一个粒子向全局最优粒子学习,其余各粒子向所在邻域中最优粒子学习,而全局最优粒子利用无速度项的简化 PSO 进化. 在此过程中,除全局最优粒子外的各粒子利用 Kalman 滤波方程更新粒子的位置. 实验表明,相较于基本 PSO 和其他 PSO 方法,即使是覆盖困难的路径,本文方法也具有进化代数少、路径覆盖成功率高及性能稳定的特点.

关键词: 测试数据生成; 粒子群优化; Kalman 滤波; 邻域拓扑
中图分类号: TP311.5 **文献标识码:** A **文章编号:** 0372-2112 (2017)10-2473-11
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.10.023

A Test Data Generation Method Based on Kalman Filter and Particle Swarm Optimization Algorithm

XUE Meng[1], JIANG Shu-juan[1,2], ZHANG Zheng-guang[1], QIAN Jun-yan[2], ZHANG Yan-mei[1,3], CAO He-ling[4]
(1. *School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangsu* 221116, *China*;
2. *Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, Guangxi* 541004, *China*;
3. *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu* 210093, *China*;
4. *College of Information Science and Engineering, Henan University of Technology, Zhengzhou, Henan* 450001, *China*)

Abstract: A test data generation method named multi-neighborhood Kalman filter PSO (MNKFPSO) was proposed to reduce the evolution number and to improve the success rate of path coverage. Particles except the global best one update themselves' positions using Kalman filter. One of them is allotted to a fixed neighborhood. A designated particle learns from the global best particle, others learn from the best in one neighborhood. And the global best particle's position changes by a simple PSO which discards the particle velocity. The experimental results show that it can generate test data covering the specified path in the less evolutionary using MNKFPSO and has high success rate of path coverage even though the paths difficult to cover. The algorithm also exhibits a stable performance.

Key words: test data generation; particle swarm optimization (PSO); Kalman filter; neighborhood topology

1 引言

测试数据生成是软件测试的一个重要环节. 路径覆盖是一种较为常用的覆盖,即在被测程序的参数输入域内寻找能使程序按照指定路径执行的数据,是软件单元测试中的一个基本问题[1].

Harman 等[2]于 2001 年首次提出基于搜索的软件工程(Search-Based Software Engineering, SBSE)概念,尝试使用一些元启发式搜索算法解决软件工程领域的问题. 基于搜索的测试数据生成技术是 SBSE 领域的一个重要分支,受到了研究人员的广泛关注,成为软件测试

收稿日期:2016-08-07;修回日期:2017-04-23;责任编辑:孙瑶
基金项目:国家自然科学基金(No. 61502497, No. 61562015, No. 61673384, No. 61602154);中国博士后科学基金(No. 2015M581887);广西可信软件重点实验室研究课题(No. KX201530);南京大学计算机软件新技术国家重点实验室开放课题(No. KFKT2014B19);徐州市科技计划项目(No. KC15SM051);河南省高等学校重点科研项目计划资助(No. 16A520005)

数据生成领域极为活跃的一个研究方向[3~6]. 该技术将测试数据生成问题通过构造适应度函数转化成数学函数优化问题,利用遗传算法(Genetic Algorithm, GA)[7~16]、粒子群优化(Particle Swarm Optimization, PSO)[17~23]算法等元启发式搜索算法寻找最优解,生成满足相应覆盖要求的测试数据.

Monson 和 Seppi[24] 从关注粒子的运动出发,运用 Kalman 滤波更新粒子的位置,提出了 Kalman 滤波和 PSO 相结合的 KSwarm 算法. 在 Kalman 滤波过程中,观测值直接使用粒子邻域内的最优解,是以该算法的收敛速度较快,需要的进化代数较少,在基准函数优化中表现了优越的性能[24]. 但是,作者在文中并没有对滤波器的构造进行深入的探讨和分析. 当我们尝试将其应用到测试数据生成中时,发现相对于基本 PSO 算法,KSwarm 算法在测试数据生成问题上的性能提升并没有基准函数优化表现的显著. 经过分析和实验验证,本文提出了应用于测试数据生成的多邻域 Kalman 滤波 PSO(Multi-Neighborhood Kalman Filter PSO, MNKFPSO)算法. 实验表明:MNKFPSO 算法在具有高路径覆盖成功率的同时,需要的进化代数较基本 PSO 和其他 PSO 算法明显减少,具有极大的优势.

2 相关概念

2.1 PSO 算法

PSO 算法最早是由 Kennedy 和 Eberhart[25] 于 1995 年提出的一种基于群体智能的随机启发式搜索算法. 本文所述基本 PSO 是指如式(1)(2)所示的含惯性权重[26]的方法.

设种群中含有的粒子数目为 N,搜索空间维度为 D,第 i 个粒子位置 $x_i=(x_{i1},x_{i2},\cdots,x_{iD})$,速度 $v_i=(v_{i1},v_{i2},\cdots,v_{iD})$,各粒子历史上搜到的最优位置为 $p_i=(p_{i1},p_{i2},\cdots,p_{iD})$. 整个种群历史上搜索到的最优位置为 $g=(g_1,g_2,\cdots,g_D)$,即全局最优解. 当种群进化到第 t 代时,粒子的速度和位置更新公式如式(1)(2)所示.

$$v_i^t=\omega v_i^{t-1}+c_1r_1(p_i^{t-1}-x_i^{t-1})+c_2r_2(g^{t-1}-x_i^{t-1}) \quad (1)$$

$$x_i^t=x_i^{t-1}+v_i^t \quad (2)$$

其中学习因子 c_1 和 c_2 均为常数,r_1 和 r_2 是(0,1)之间均匀分布随机数,惯性权重 ω 的取值一般位于[0.2, 0.9].

式(1)中,g^{t-1}代表种群所有粒子截止 $t-1$ 代曾搜索到的最优位置,因此也称为全局版的 PSO. 当使用粒子 i 所在邻域的所有粒子曾搜索到的最优位置 Lb^{t-1}代替 g^{t-1}时,称为局部版的 PSO.

2.2 Kalman 滤波

Kalman 滤波[27]是 Kalman 在 1960 年第一次提出的一种线性最小方差估计理论. 在航天、计算机图像处理、目标跟踪等领域得到了广泛应用.

Kalman 滤波算法首先使用式(3)(4)的时间更新方程,根据上一时刻的后验估计值来估计当前时刻的状态,得到当前状态的先验估计值;然后再使用当前时刻的测量值通过式(5)(6)(7)的测量更新方程来修正这个估计值,得到当前时刻的后验估计值,即当前时刻的最优估计值.

离散 Kalman 滤波的时间更新方程(不考虑控制作用及系统过程噪声):

$$\boldsymbol{X}_{t|t-1}=\boldsymbol{A}\boldsymbol{X}_{t-1} \quad (3)$$

$$\boldsymbol{P}_{t|t-1}=\boldsymbol{A}\boldsymbol{P}_{t-1}\boldsymbol{A}^{\mathrm{T}}+\boldsymbol{Q} \quad (4)$$

离散 Kalman 滤波的测量更新方程:

$$\boldsymbol{K}_t=\boldsymbol{P}_{t|t-1}\boldsymbol{H}^{\mathrm{T}}(\boldsymbol{H}\boldsymbol{P}_{t|t-1}\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{R})^{-1} \quad (5)$$

$$\hat{\boldsymbol{X}}_t=\boldsymbol{X}_{t|t-1}+\boldsymbol{K}_t(\boldsymbol{Z}_t-\boldsymbol{H}\boldsymbol{X}_{t|t-1}) \quad (6)$$

$$\boldsymbol{P}_t=(\boldsymbol{I}-\boldsymbol{K}_t\boldsymbol{H})\boldsymbol{P}_{t|t-1} \quad (7)$$

其中:$\boldsymbol{X}_{t-1}$为 $t-1$ 时刻的系统状态变量;$\boldsymbol{X}_{t|t-1}$为状态一步预测变量;$\boldsymbol{A}$ 为状态转移矩阵;$\boldsymbol{P}_{t-1}$、$\boldsymbol{P}_t$ 为 $t-1$ 和 t 时刻的滤波均方误差;$\boldsymbol{P}_{t|t-1}$为一步预测均方误差;$\boldsymbol{Q}$ 为过程激励噪声协方差矩阵,用于表示状态转移矩阵与实际过程之间的误差;$\boldsymbol{H}$ 为系统观测矩阵;$\boldsymbol{K}_t$ 为 Kalman 滤波增益;$\boldsymbol{R}$ 为测量噪声协方差矩阵;$\boldsymbol{Z}_t$ 为观测值,联合滤波增益 $\boldsymbol{K}_t$ 对一步预测值 $\boldsymbol{X}_{t|t-1}$进行修正,以得到最优估计值 $\hat{\boldsymbol{X}}_t$.

2.3 邻域拓扑

粒子的邻域拓扑是指邻域内粒子之间连接和信息共享的方式,决定了粒子的运动轨迹受哪些粒子信息影响. 在静态邻域拓扑中,粒子的数目是固定的;而动态邻域拓扑中的粒子是动态变换的. 常见的邻域拓扑结构有:全互联结构、环形结构、冯诺伊曼结构、随机拓扑[28]等. 不同的邻域拓扑结构对 PSO 算法性能有较大影响[29],但是最佳效果的拓扑结构和所选择的基准测试函数有密切关系.

粒子被划分到各邻域一般有两种方法:一是根据粒子的编号,按照邻域拓扑结构进行划分;二是按照粒子之间的欧氏空间距离进行划分[30],该方法虽然经过实验效果较好,但计算量太大,且需要大量存储空间,一般不使用. 本文多邻域拓扑采用第一种依据粒子编号划分的方法.

3 基于 Kalman 滤波和 PSO 的测试数据生成方法

3.1 KSwarm 算法的改进

经过分析及实验验证,为使 KSwarm 算法能有效应用于测试数据生成,做如下两点改进:

① 在原算法中,邻域拓扑采用"start"[31]结构,为了能获得更快的收敛速度,本文设计了多邻域拓扑结构

及相应的粒子学习策略,详细内容在 3.2 节中介绍.

② 在原算法中,观测值采用式(8)(9)产生,为防止陷入局部最优:$\phi \in [0,2)$. Lb 为邻域内所有粒子的最优位置,式(8)z_v 的计算表明邻域内的粒子快速向该邻域的最优位置靠拢.

$$z_v = \phi(Lb - x) \tag{8}$$

$$z_p = x + z_v \tag{9}$$

在此,将原算法中定义的观测向量:$\boldsymbol{Z} = (z_p, z_v)^{\mathrm{T}}$,更改为:$\boldsymbol{Z} = (z_p)^{\mathrm{T}}$. 主要原因是:需要生成的测试数据是由粒子的位置信息表示的,去掉 z_v 还可以简化矩阵运算. 此时,粒子 i 在进化到 t 代时,系统观测方程如式(10)所示(在此不考虑观测噪声,$\boldsymbol{H}$ 为系统观测矩阵).

$$\boldsymbol{Z}_i^t = \boldsymbol{H}\boldsymbol{X}_i^t = \boldsymbol{H}\begin{bmatrix} z_{pi}^t \\ z_{vi}^t \end{bmatrix} = \boldsymbol{H}\begin{bmatrix} x_i^{t-1} + \phi(Lb^{t-1} - x_i^{t-1}) \\ \phi(Lb^{t-1} - x_i^{t-1}) \end{bmatrix} \tag{10}$$

式(3)~(7)的 Kalman 滤波方程各参数取值如下(假设搜索空间维度为 D,$\boldsymbol{I}_D$ 为 $D \times D$ 的单位矩阵):$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_D & \boldsymbol{I}_D \\ 0 & \boldsymbol{I}_D \end{bmatrix}$; $\boldsymbol{H} = [\boldsymbol{I}_D \quad 0]$; $\boldsymbol{X}_{t-1} = \begin{bmatrix} x_i^{t-1} \\ v_i^{t-1} \end{bmatrix}$; $\boldsymbol{P}_{t-1}$、$\boldsymbol{P}_t$ 为 $2D \times 2D$ 大小的矩阵,初始 $\boldsymbol{P}_0$ 主对线取非零值,其余元素取 0 值,为方便取 $\boldsymbol{P}_0 = \boldsymbol{I}_{2D\times 2D}$; $\boldsymbol{Q} = \boldsymbol{I}_{2D\times 2D}$;$\boldsymbol{R} = \boldsymbol{I}_{D\times D}$.

将 $\boldsymbol{X}_i^{t-1}$ 及观测值 $\boldsymbol{Z}_i^t$ 代入式(3)~(7)后,得出估计值 $\hat{\boldsymbol{X}}_i^t$. 根据 Kalman 滤波理论,在滤波"稳态"状态下,下一状态可以由式(11)得出.

$$\boldsymbol{X}_i^t \sim \mathrm{Normal}(\boldsymbol{A}\hat{\boldsymbol{X}}_i^t, \boldsymbol{P}_i^t),$$
$$\boldsymbol{P}_i^t \text{ 取主对角线元素} \tag{11}$$

3.2　多邻域拓扑划分及粒子学习策略

为尽可能避免陷入邻域内局部最优,在不增加算法复杂性的前提下,设计一种简单实用的多邻域拓扑结构及粒子学习策略:将粒子按连续编号均匀分到各邻域内,在各邻域内指定一个粒子向全局最优位置学习,其余粒子(拥有整个种群全局最优位置的粒子除外)采用全连接结构. 指定向全局最优位置学习的粒子可以采用静态方式指定,即从始至终是一固定粒子接收全局最优位置信息,也可以动态随机指定邻域内的一个粒子. 为方便处理,本文采用静态方式.

设种群中含有的粒子数目为 N,编号为$(0,1,\cdots,N-1)$,需要划定的邻域数目为 L,编号为$(0,1,\cdots,L-1)$. 为保证公平及计算简便,各邻域内粒子数目保持一致,选择 $L|N$. 此时,每个邻域内粒子数目为 $M = N/L$.

设 $i \in [0, N-1]$,$l \in [0, L-1]$,数组 $MNT[L][2]$存储粒子划分情况,编号在$[M*l, M*(l+1)-1]$范围内粒子将被分配到编号为 l 邻域内. 在邻域 l 中,指定编号 $MNT[l][0]$的粒子用种群全局最优位置更新.

多邻域拓扑划分方法如算法 1 所示.

算法 1　多邻域拓扑划分方法

```
输入:种群粒子数目 N,邻域数目 L
输出:邻域划分信息存储二维数组 MNT
1 begin
2   MNT = new int[L][2];
3   M = N/L;
4   for l = 0 to L - 1 do
5       MNT[l][0] = M * l;
6       MNT[l][1] = M * (l + 1) - 1;
7   endfor
8 end
```

结合 3.1 节所述内容,改进的 KSwarm 算法如算法 2 所示.

算法 2　改进的 KSwarm 算法

```
输入:粒子编号 i,粒子 i 在 t-1 代的位置 x_i^{t-1} 和速度 v_i^{t-1},二维数组
      MNT[][2],种群粒子数目 N,邻域数目 L
输出:粒子 i 在 t 代的位置 x_i^t 和速度 v_i^t
1 begin
2   M = N/L; l = [i/M];
3   if (i <> MNT[l][0]) then
4       在邻域 l 内寻找拥有历史局部最优位置的粒子 lbesti
5       Lb^{t-1} = 粒子 lbesti 的历史最优位置 p_lbesti
6   else
7       Lb^{t-1} = 种群历史全局最优位置 g
8   endif
9   计算观测向量 Z_i^t = x_i^{t-1} + Random(0,2) × (Lb^{t-1} - x_i^{t-1})
10  将位置 x_i^{t-1} 和速度 v_i^{t-1} 构成列向量 X_i^{t-1}
11  将 X_i^{t-1} 和 Z_i^t 代入式(3)~(7)计算 X̂_i^t,并保存 P_i^t
12  根据式(11)计算粒子 i 在 t 代的位置 x_i^t 和速度 v_i^t
13 end
```

从算法 1、2 可以看出,本文方法将整个种群粒子分成了三类:一是一个拥有种群历史全局最优解的粒子,编号记为 gbesti;二是各邻域中指定的编号最小的粒子;三是其他粒子. 这三类粒子中,除了唯一的一个粒子 gbesti 外,其余粒子均有学习的对象. 因此,粒子 gbesti 需要额外单独处理,以引导整个种群进化.

对于粒子 gbesti,本文使用文献[32]提出的去掉速度项的简化 PSO 计算其位置,位置更新公式如式(12)所示,速度更新采用式(13)进行.

$$x_{\mathrm{gbesti}}^t = \omega x_{\mathrm{gbesti}}^{t-1} + c_1 r_1 (p_{\mathrm{gbesti}}^{t-1} - x_i^{t-1}) + c_2 r_2 (g^{t-1} - x_{\mathrm{gbesti}}^{t-1}) \tag{12}$$

$$v_{\mathrm{gbesti}}^t = x_{\mathrm{gbesti}}^t - x_{\mathrm{gbesti}}^{t-1} \tag{13}$$

参数 c_1、c_2、r_1、r_2、ω 的含义和式(1)参数说明相同.

3.3 多邻域 Kalman 滤波 PSO 测试数据生成方法

3.3.1 MNKFPSO 算法

综合 3.1、3.2 节内容,MNKFPSO 算法完整描述如算法 3 所示. 在算法 3 中,3 ~ 9 行为粒子群状态初始化,和基本 PSO 算法中的做法相同;11 ~ 22 行根据得到的邻域拓扑结构对三类粒子分别进行处理. 停机条件是:找到测试数据或达到最大进化代数.

算法 3 多邻域 Kalman 滤波 PSO(MNKFPSO)算法

```
1 begin
2   初始化种群粒子数目 N,邻域数目 L(L<N);最大进化代数 T_max,
    最大速度 V_max,参数取值范围[x_min,x_max],惯性权重 ω 等参数;根
    据 3.1 节,初始化式(3) ~ (7)中各参数矩阵
3   初始化各粒子的位置和速度
4   for each particle i do
5       计算适应度值 fitnessi
6       粒子自身局部最优位置 = 初始位置;
7       粒子自身初始最优适应度值 = fitnessi
8   endfor
9   搜寻种群全局最优适应度值及对应位置,并保存,相应的粒子编
    号记为 gbesti
10  用算法 1 进行多邻域划分
11  While(不满足停机条件)
12      for each particle i do
13          if (i = = gbesti) then
14              使用式(12)(13)更新其位置和速度
15          else
16              用算法 2 处理
17          endif
18          粒子位置及速度边界变异(见 4.3 节算法 4)
19          计算粒子适应度值,更新各粒子自身局部最优解
20      endfor
21      更新种群全局最优解、适应度值及 gbesti
22  endWhile
23  输出结果信息
24 end
```

3.3.2 基于 MNKFPSO 算法的测试数据生成模型

基于 MNKFPSO 算法的测试数据生成模型如图 1 所示,包含测试环境构造及预处理、待测程序运行和 MNKFPSO 算法 3 个模块:

① 测试环境构造及预处理模块完成如下工作:对源程序生成控制流程图、进行静态分析、对程序进行插桩并选择目标路径、提取有用的参数作为算法模块的输入.

② 待测程序运行模块:利用算法模块传递进来的测试数据驱动待测程序运行,进而计算出适应度值反馈给算法模块. 本文实验采用的适应度函数构造方法在第 4.2 节中叙述.

③ 算法模块是本模型的核心部分,边界变异策略如算法 4 所述(4.3 节),其余在前文已详细叙述,在此不再赘述.

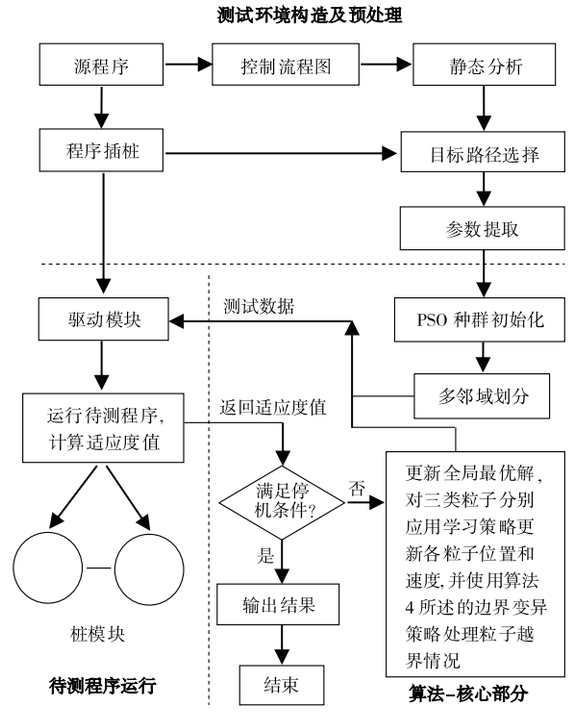


图1 基于MNKFPSO算法的测试数据生成模型

4 实验及结果分析

4.1 实验对象

实验选取了 10 个被广泛使用的基准被测程序,如表 1 所示. 表中代码行数和圈复杂度由 JavaNCSS[33] 获得.

程序 1 的目标路径是数组中所有的元素均为 0;本文将程序 2 两个数组的长度均设为 10,目标路径是两个数组中相对应位置的元素均相等;对程序 3 稍作修改,目标路径为:输入的 10 个字符均为 16 进制的字符,且其转化为 10 进制之后的总和在 50 ~ 100 之间;将程序 4 号码长度固定为 16,目标路径为每个号码均在 0 ~ 9 之间,且满足 luhnCheck 函数的条件;程序 5 的目标路径是二次方程式满足 $b^2-4ac=0$ 且 $a\neq 0, c\neq 0$;程序 6 的目标路径是年月日满足 1912 ~ 2050 年间的闰年 2 月 29 日;程序 7 的目标路径为等边三角形;程序 8 的目标路径是某一随机变量 x 等于中位数 c;程序 9 固定其行列数为 2 和 8;程序 10 的 begin( )方法第一个参数也由搜索算法产生. 在程序 9 中随机选择一条目标路径,在程序 10 中选择两条目标路径分别作为单独路径进行测试数据生成. 在程序 10 的这两条目标路径中,一条目标路径要求其中一个参数有唯一解,另一条路径中有三

个参数取值唯一,为方便,下文称为“程序 11”. 在选择的目标路径中:程序 2、3、4、9 由于均带有循环且参数个数较多而较难覆盖;程序 5 取值范围大且逻辑结构复杂,覆盖目标路径难度较大;程序 6 目标路径的成功覆盖需要其中两个参数取值唯一;程序 11 选择的目标路径不仅约束多,且输入参数间存在相互制约,含有多处“==”精确值判断,13 个变量中有 3 个变量在搜索空间中只有唯一取值. 这些程序变量个数不等,圈复杂度差异较大,选取目标路径中的分支含有“==”、“&&”、“||”等多种表达式,具有较强的代表性.

表 1　用于实验的被测程序

编号	名称	变量个数	代码行数	圈复杂度	来源
1	all _ zeros	20	13	5	文献[34]
2	Arrays (Equals)	20	13	11	Java Collections
3	change16to10	10	112	29	文献[35]
4	CreditCardValidator (luhnCheck)	16	22	7	Apache Commons Valid
5	Quadratic	3	39	15	文献[36]
6	NextDate	3	55	18	文献[36]
7	Triangle	3	23	4	文献[35]
8	TriangularDistribution (density)	4	84	49	Apache Commons Math
9	Tot _ info	16	281	122	西门子套件
10	jtcas	13	132	44	西门子套件

4.2　适应度函数设计

适应度函数构造时使用分支距离法[37],即在各个分支节点都插入分支函数 f_i,用于表示当前测试数据与该分支为真的距离. 若 f_i 小于等于 0,则表明覆盖了该分支,将 f_i 设为 0. 若该目标路径含有 m 个分支节点,则适应度函数 Fitness 计算方法如式(14)所示:

$$\text{Fitness} = \frac{\sum_{i=0}^{m-1} \frac{1}{f_i + 1}}{m} \times 100 \tag{14}$$

4.3　边界变异及参数设置

边界变异策略有很多[38],如:吸收墙、反射墙、隐匿墙、衰减墙、随机速度和对搜索空间进行缩放等. 由于边界值变异不是本文重点内容,所以采用如算法 4 所示的较为简单的变异策略.

算法 4　边界变异

```
1 begin
2   if x_{i,d} > x_max then
3     x_{i,d} = x_max - 0.5 × Random(0,1) × (x_max - x_min);
4   endif
5   if x_{i,d} < x_min then
6     x_{i,d} = x_max + 0.5 × Random(0,1) × (x_max - x_min);
7   endif
8   if v_{i,d} > v_max then
9     v_{i,d} = v_max - 0.5 × Random(0,1) × v_max;
10  endif
11  if v_{i,d} < -v_max then
12    v_{i,d} = -v_max + 0.5 × Random(0,1) × v_max;
13  endif
14 end
```

最大进化代数 $T_{\max} = 10000$. PSO 算法中的参数 ω,使用如式(15)所示的线性方式变化,其中,Fitness_i^{t-1} 为粒子 i 在 $t-1$ 代的适应度函数值.

和边界变异类似,参数设置也非本文重点,所以在此不做特别讨论、优化和处理.

$$\omega^t = 0.8 - 0.5 * \text{Fitness}_i^{t-1}/100 \tag{15}$$

4.4　实验设计及结果度量指标

实验中对 MNKFPSO 算法和其他 PSO 算法在适应度函数、边界变异策略和参数设置上相同.

实验采用如下方式进行:对各算法做重复 1000 次实验,记录每次实验是否成功搜索到测试数据、成功搜索到测试数据时所需的进化代数和每次运行各搜索算法花费的时间.

实验平台配置为:Win7 系统、6 代酷睿 I5 CPU 3.2G 主频、8G 内存. 为对比各算法在测试数据生成方面的搜索效果,使用如下 3 个度量指标进行分析. 一是路径覆盖成功率(Success Rate,SR):成功搜索到覆盖目标路径测试数据的次数所占比例;二是平均进化代数(Average Generation,AG):成功搜索到覆盖目标路径测试数据时的平均进化代数;三是平均运行时间(Average Running Time,ART):实验运行算法花费时间的平均值,单位 ms.

4.5　实验结果及分析

4.5.1　本文方法与基本 PSO、OL-PSO 方法的比较

MNKFPSO 与基本 PSO、OL-PSO[22]方法(基于奇异值分解的正交搜索和局部搜索策略 AVM(Alternating Variable Method)相结合的 PSO)关于 SR、AG 和 ART 的实验结果对比如表 2 所示. MNKFPSO 算法中邻域数目 $L = 10, N = 50$.

表 2 三种算法关于覆盖成功率、平均进化代数和平均运行时间的结果

程序编号	最大进化代数 $T_{max}=1000$									最大进化代数 $T_{max}=10000$								
	基本 PSO			OL-PSO			MNKFPSO			基本 PSO			OL-PSO			MNKFPSO		
	SR%	AG	ART (ms)	SR%	AG	ART (ms)	SR%	AG	ART (ms)	SR%	AG	ART (ms)	SR%	AG	ART (ms)	SR%	AG	ART (ms)
1	**87.7**	816.60	25.677	100	22.66	2.34	100	16.02	3.027	100	**848.84**	25.647	100	22.66	2.34	100	16.02	3.027
2	**99.4**	163.76	5.242	100	**3.09**	0.266	100	**8.54**	2.043	**99.80**	180.50	5.616	100	**3.09**	0.266	100	**8.54**	**2.043**
3	**93.3**	111.81	**3.541**	99.8	171.78	10.374	100	125.22	**12.746**	**97.70**	233.44	**9.703**	100	173.55	9.999	100	125.22	**12.746**
4	100	124.53	**5.054**	**94.2**	**379.51**	52.212	100	4.35	0.655	100	**124.53**	5.054	100	**426.40**	52.916	100	**4.35**	**0.655**
5	**76.1**	206.34	**4.477**	**7.9**	**483.89**	**71.214**	**95.4**	35.30	2.808	**92.90**	750.10	**16.567**	**51.1**	**3859.4**	**73.008**	**97.2**	94.60	**17.721**
6	100	**20.06**	0.281	100	17.25	0.219	100	**28.90**	**1.014**	100	**20.06**	0.281	100	17.25	0.219	100	**28.90**	**1.014**
7	100	55.37	0.78	100	56.58	0.671	100	25.47	**0.905**	100	55.37	0.78	100	56.58	0.671	100	25.47	0.905
8	100	22.22	0.265	100	6.66	0.093	100	6.27	**0.281**	100	22.22	0.265	100	6.66	0.093	100	6.27	0.281
9	100	4.07	14.059	100	2.58	27.052	100	1.48	6.756	100	4.07	14.059	100	2.58	27.052	100	1.48	6.756
10	100	16.50	0.577	100	18.52	1.529	100	12.76	1.576	100	16.50	0.577	100	18.52	1.529	100	12.76	1.576
11	**99.3**	**86.22**	**2.256**	**93.9**	285.93	31.356	**98.6**	**203.66**	**23.36**	**99.4**	**95.01**	**3.468**	100	391.46	37.066	100	**223.83**	**24.371**

从表 2 可以看出,无论最大进化代数设置为 1000 还是 10000,本文 MNKFPSO 算法在 AG 和 SR 两评价指标上,除程序 6、11 之外的测试程序都表现出了远优于基本 PSO 的性能. 在 $T_{max}=10000$ 时,程序 1 的平均进化代数从 848.84 下降到 16.02,高达 52 倍;程序 3 降低略小些,仅 46% 左右;程序 6 和 11 是两个 AG 略有升高的程序,主要原因是程序 6 搜索的是闰年 2 月 29 日,搜索目标是一精确值,程序 11 搜索的结果中需要三个参数分量取值唯一. 可以看出本文方法在精确值搜索上较基本 PSO 稍弱. 若能结合 OL-PSO 算法中 AVM 方法进行局部搜索相信可以加快速度. OL-PSO 即使使用了 AVM 局部搜索方法,也仅在程序 2、6 的 AG 指标上略优于本文方法. 在覆盖成功率方面,本文算法明显要优于其他两种算法,尤其是程序 5 较难覆盖的目标路径,本文方法的 SR 依然达到了 97.2%. 当 $T_{max}=1000$ 时,本文算法在 AG 和 SR 评价指标上较基本 PSO 和 OL-PSO 的优势极其显著,尤其是 SR 指标,本文算法在除程序 5、11 外的其他程序上均能达到 100%,而基本 PSO 和 OL-PSO 在程序 1、2、3、4、11 上出现了不同程度的下降. 尤其对于程序 5,变量搜索范围大且目标路径逻辑复杂、难以覆盖,但本文算法也仅下降了 1.8%,基本 PSO 下降了 16.8%,而 OL-PSO 下降高达 43.5%. OL-PSO 算法之所以对程序 5 的 SR 很低,是因为其中使用 AVM 方法搜索到次优解致使种群的多样性降低,造成总体覆盖成功率较低,效果较基本 PSO 还差[22]. 在 ART 指标上:三种方法在不同程序上表现不同,没有任何一种方法明显优于其他方法,即三种方法在 ART 方面的性能优势和实际的目标路径有关.

比较 T_{max} 为 10000 和 1000 时的结果,基本 PSO 和 OL-PSO 都出现了多个程序 SR 指标一定程度的下降,这恰恰说明了 T_{max} 的设置对测试数据生成效率有一定的影响,也说明了降低进化代数的必要性.

为进一步说明本文方法在测试数据生成进化代数上的优势及有效性,对实验结果进行非参数统计检验和效应量分析. 表 3 是利用 Mann-Whitney U 检验和 Cohen's d 效应量对各算法进行分析的结果. $T_{max}=10000$, $N=50$.

表 3 中的 Mann-Whitney U 统计检验,假设分别为基本 PSO、OL-PSO 的进化代数与本文方法差异不显著. 对基本 PSO 和 MNKFPSO 方法而言,所有程序的 p-value 均小于 0.05,拒绝原假设,说明本文方法的进化代数与基本 PSO 方法存在显著性差异. 相对 OL-PSO 方法,只有程序 3 的 p-value 为 $0.194>0.05$,说明对于程序 3 本文方法与 OL-PSO 方法差异不显著,而其他的 10 个程序均能拒绝原假设. 效应量分析方面:基本 PSO 和 MNKFPSO 比较,程序 1、2、4、7、8、9 效应量值大于 0.8,说明效应明显,样本分布重叠度小,MNKFPSO 比基本 PSO 方法进化代数小;OL-PSO 和 MNKFPSO 比较,程序 1、2、4、5、7 的效应量大于 0.8 说明效应明显. 程序 6 的效应值分别为 -0.32、-0.42,说明效应中等,MNKFPSO 方法的进化代数高于基本 PSO 和 OL-PSO 方法. 程序 3 的效应值分别为 0.21、0.36,也说明效应中等,不过 MNKFPSO 方法平均进化代数小于其他两种方法. 程序 2 的 OL-PSO 与 MNKFPSO 方法效应量达到 -6.58,说明效应明显,MNKFPSO 方法进化代数高于 OL-PSO 方法. OL-PSO 与 MNKFPSO 方法在程序 8、9 上效应量分布仅为 0.17、0.18,说明效应小,样本分布重叠度高,OL-PSO 与本文方法差异不大. 另外,表 3 中效应量绝大部分是正值,也说明了本文方法较其他两种方法具有优势. 基本 PSO 与本文方法在程序 6、11 上的效应量为

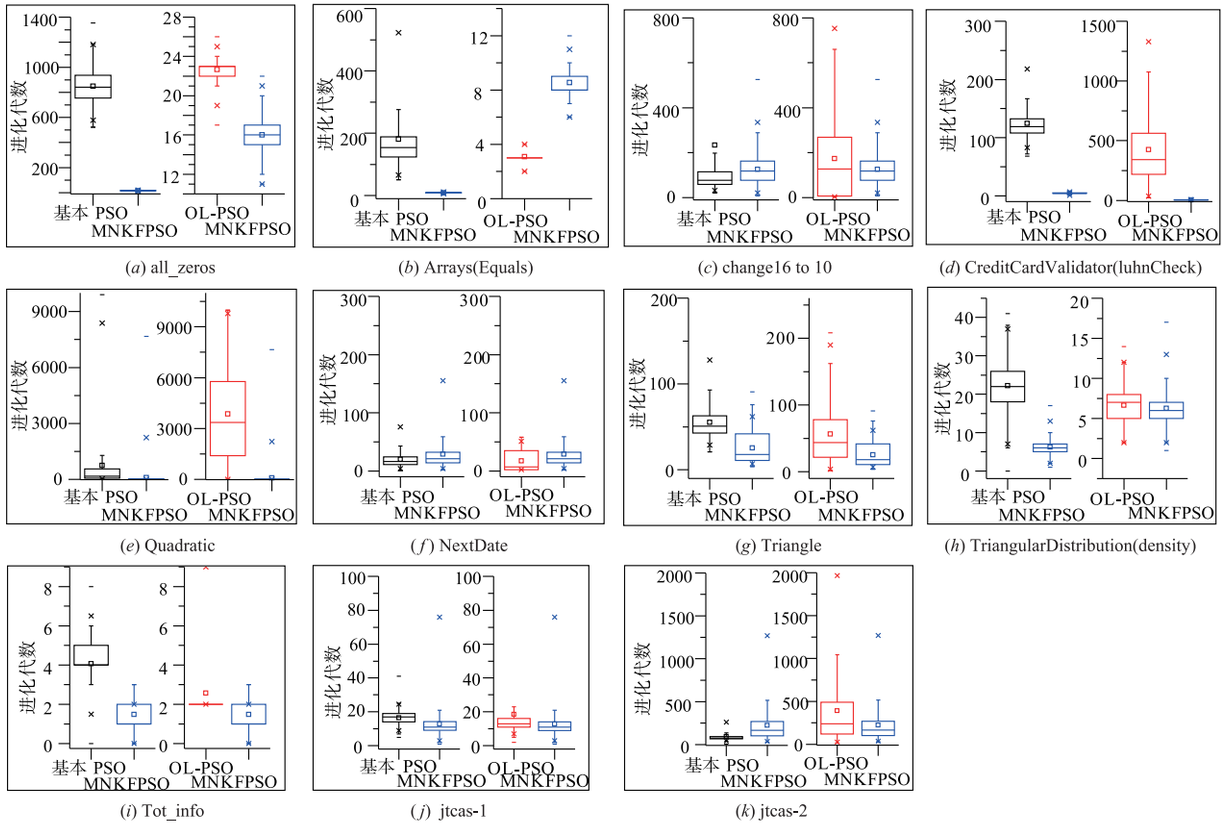
负值,也说明了本文方法对输入域中仅有唯一取值的测试数据生成效率不如基本 PSO.

表 3　三种算法关于进化代数的 p-value 值和效应量值

程序编号	基本 PSO vs MNKFPSO		OL-PSO vs MNKFPSO	
	p-value	Cohen's d	p-value	Cohen's d
1	<0.001	8.93	<0.001	4.05
2	<0.001	0.81	<0.001	**-6.58**
3	<0.001	**0.21**	0.194	**0.36**
4	<0.001	6.08	<0.001	2.06
5	<0.001	**0.58**	<0.001	1.91
6	<0.001	**-0.32**	<0.001	**-0.42**
7	<0.001	1.62	<0.001	0.92
8	<0.001	3.44	<0.001	**0.17**
9	<0.001	3.25	<0.001	**0.18**
10	<0.001	**0.41**	<0.001	**0.35**
11	<0.001	**-0.54**	<0.001	**0.46**

4.5.2　进化代数稳定性分析

从两个方面进行进化代数稳定性分析:一是分析各次实验结果是否平稳;二是观察在修改算法一些关键参数情况下,实验结果是否波动较大. 图 2 是三种算法收敛代数的分布情况. 数据是 1000 次实验的结果($N=50, L=10$),图 2 显示了在 T_{max} 内成功搜索到测试数据时实际使用的进化代数分布情况. 从图中可以看出,除程序 3、6、11 外,MNKFPSO 算法的进化代数中位数都明显大幅低于基本 PSO,但是程序 3 的数据是剔除了 23 个达到 T_{max} 数据之后的结果,就 AG 而言,MNKFPSO 具有一定的优势. 除程序 2、6 外,MNKFPSO 算法的进化代数中位数均低于 OL-PSO. 同时,也可从各图最小值至最大值、下四分位数至上四分位数表示的区间看出,除程序 3、6、7、11 外,本文方法波动范围明显小于基本 PSO 方法;除程序 1、2、9 外,本文方法进化代数波动范围也优于 OL-PSO 方法. 总体上,本文方法进化代数的波动范围较小、性能平稳、优势明显,在生成测试数据时,其进化代数较为稳定.



(*a*) all_zeros　(*b*) Arrays(Equals)　(*c*) change16 to 10　(*d*) CreditCardValidator(luhnCheck)

(*e*) Quadratic　(*f*) NextDate　(*g*) Triangle　(*h*) TriangularDistribution(density)

(*i*) Tot_info　(*j*) jtcas-1　(*k*) jtcas-2

图2　进化代数的分布情况

另外一个需要重点考查的参数是种群大小. 本文在实验中统一对种群大小进行变化,观察对 AG 和 SR 的影响. 表 4、图 3 和图 4 分别显示了种群大小 N 与 SR、AG 及 ART 之间的关系($L=10$). 由于除程序 5、11 之外其他程序的 SR 在不同种群大小时均为 100%,所以表 4 仅列出程序 5、11 的情况.

表 4 种群大小(N)与覆盖成功率(SR%)之间的关系

程序编号	$N=50$	$N=60$	$N=70$	$N=80$	$N=90$	$N=100$
5	97.20	96.7	97.4	97.3	96.5	97.6
11	100	99.7	99.9	99.7	99.9	99.7

从表 4 可以看出,虽然程序 5、11 的 SR 随 N 的增大存在一定的变化,但变化并不明显. 程序 5 在 $N=100$ 时达到最大覆盖成功率,但也仅比 $N=50$ 时多 0.4%,比最低的 $N=90$ 时 96.5% 的覆盖成功率也仅高 1.1%,随着粒子数目 N 从 50 增加到 100,覆盖成功率虽存在反复波动,但不存在剧烈升降现象. 程序 11 在 $N=50$ 时 SR 是 100%,随着 N 的增加,SR 反而无法达到 100%. 可见 SR 并不是随着 N 增加就一定有所提高.

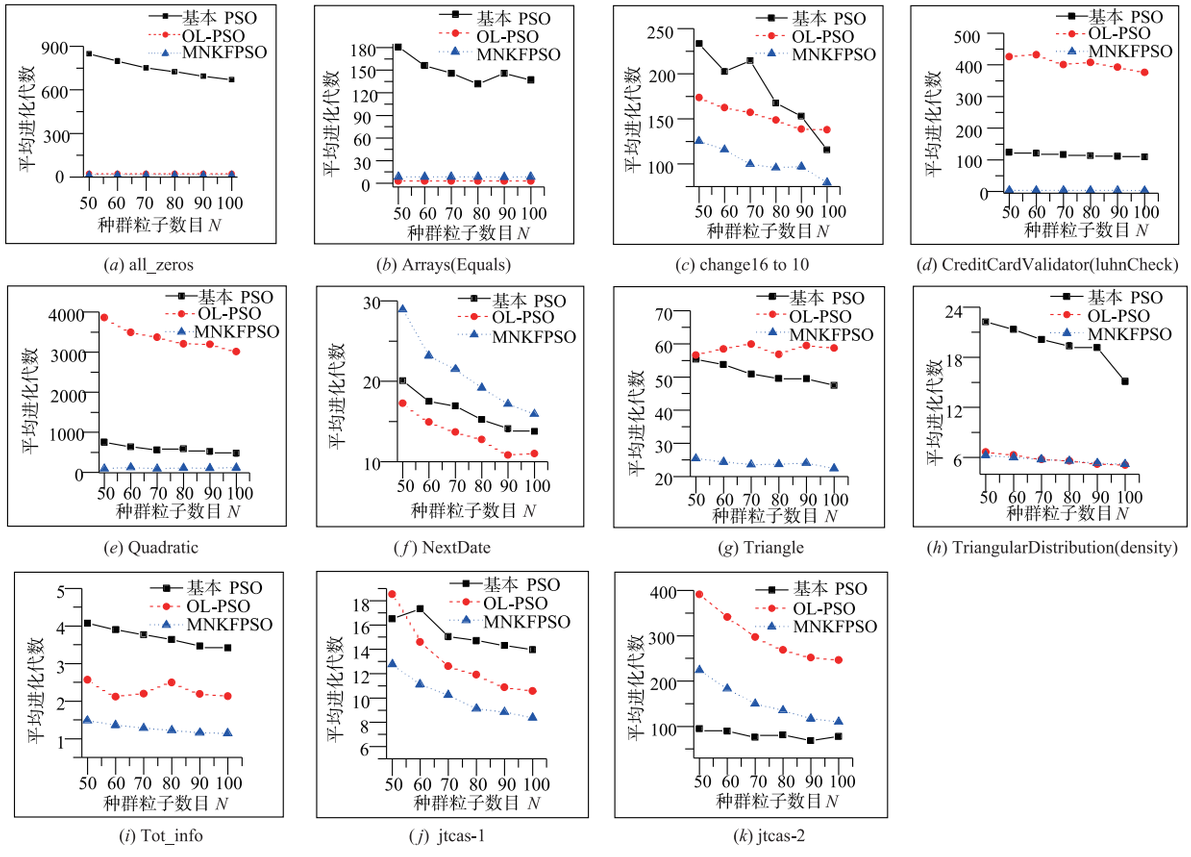


图3 种群大小与平均进化代数之间的关系

从图 3 可以看出:基本 PSO 算法生成测试数据的 AG 随 N 的增大变化较大,趋势是随着 N 增加,进化代数减少,在 N 增加到 100 后,AG 比 $N=50$ 时都有不同程度降低. 但是,程序 2、3、10、11 的 AG 随着 N 增大,其间会出现上下波动现象. 基本 PSO 算法需要在进化代数和 N 之间进行性能平衡. 对程序 7,OL-PSO 算法随着 N 增加,AG 整体趋势是增加的;对程序 9,OL-PSO 算法随 N 增加上下波动,尽管波动范围不大. 对 MNKFPSO 算法而言,在生成测试数据时,除程序 6、10、11 外,AG 基本平稳,并没有随 N 的变化而出现较明显变化. 即使是程序 6、10、11,进化代数也是随着 N 的增大而减少,这三个程序选定的目标路径中均有输入取值唯一情况,本文算法对此表现的性能稍弱.

因此,在使用本文方法生成测试数据时,不需要提高 N 来获得更高的覆盖成功率及更少的进化代数. 这在一定程度上也降低了计算量.

图 4 显示了种群粒子数目 N 与 ART 之间的关系. 从图中看出,随着 N 的增加,几乎所有程序的时间开销都是增加的. 绝对值增幅最大的是程序 5,达到 14.6ms,最小的是程序 8,仅 0.23ms;相对增加幅度最大的是程序 7,达到 103%,最小的是程序 6 增加了 26%.

从以上分析可以看出,本文提出的 MNKFPSO 算法在生成测试数据时性能稳定.

5 相关工作

自 Xanthakis 等[7]将遗传算法应用于软件测试数据生成后,研究人员对元启发式搜索算法在其中的应用进行了大量研究和完善. 相比于 GA,PSO 算法在测试

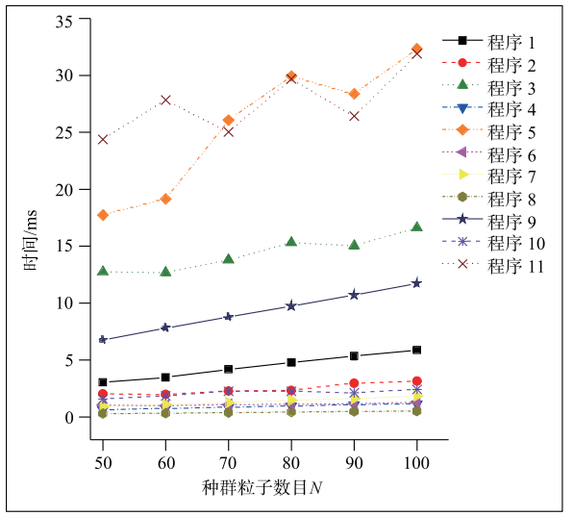


图4　种群大小与平均运行时间之间的关系

数据生成领域的研究才刚起步. 2007 年,Windisch 等[17]将 PSO 算法应用到基于结构的测试中,基于分支覆盖生成测试数据,对一些程序进行了实验研究,结果表明 PSO 算法在代码覆盖和执行效率上要优于 GA. 毛澄映等[18,19]通过对比实验研究,进一步证实了 PSO 算法在平均覆盖率、全覆盖成功率、平均收敛代数和搜索时间 4 项指标上均要优于 GA. 姜淑娟等[20-23]对 PSO 在测试数据生成中的应用进行了持续的关注和研究.

目前,在测试数据生成中应用 PSO 算法时,主要针对 PSO 算法的惯性权重等参数进行调整[20]、检测陷入局部最优并增加粒子多样性[17,22]以跳出局部最优等方面进行改进. 而王令赛等[22]通过实验表明:调节惯性权重的方法对于变量数目少且逻辑简单的程序效果显著,对于含有多个变量且逻辑复杂的程序来说,惯性权重的调节作用有限. 本文方法的出发点不同于其他方法,是将 Kalman 滤波应用到 PSO 位置更新方程中,从根本上改变粒子的质点运动,从而能够减少进化代数. 通过实验验证,发现对大部分程序而言,进化代数的减少是"巨大"的.

6　总结与展望

本文 MNKFPSO 算法是将 Kalman 滤波和粒子群优化算法相结合的方法,在应用于测试数据生成时可以在较少的进化代数内搜索到覆盖相应路径的测试数据. 通过实验证实:本文算法在保证高覆盖成功率的同时,在测试数据生成的进化代数上相较于基本 PSO、OL-PSO 具有较大优势.

MNKFPSO 算法之所以能快速收敛,主要得益于使用本文设计的多邻域拓扑结构及粒子学习策略,使得在 Kalman 滤波计算 $\hat{\boldsymbol{X}}_i^t$ 过程中,各粒子可以直接快速向最优粒子学习. 在算法执行过程中,通过式(11)、多邻域拓扑结构及粒子学习策略的处理,增加了随机性,可以在一定程度上使进化避免陷入局部最优. 在 MNKFPSO 算法中若对陷入局部最优采取预防与检测消除相结合的方法或许可以进一步加快算法的收敛,这是一个还需要研究与验证的问题. 本文算法在时间开销上没有明显优势,因此还需要进一步提升算法性能. 实验中发现本文算法对多个输入参数仅有唯一解的程序搜索性能弱于基本 PSO,可以在本文算法中引入局部搜索(如:AVM)以提升搜索效率. 另外,在 Kalman 滤波过程中,观测值仅使用了邻域最优值,并没有结合粒子自身的历史最优值,寻找这两者更好、更合理的结合方法以得出观测值也值得继续探讨.

参考文献

[1] 单锦辉,王戟,齐治昌. 面向路径的测试数据自动生成方法述评[J]. 电子学报,2004,32(1):109-113.
SHAN Jin-Hui, WANG Ji, QI Zhi-Chang. Survey on path-wise automatic generation of test data[J]. Acta Electronica Sinica, 2004, 32(1): 109-113. (in Chinese)

[2] HARMAN M, JONES B F. Search-based software engineering[J]. Information and software Technology, 2001, 43(14): 833-839.

[3] MCMINN P. Search-based software test data generation: a survey[J]. Software Testing Verification & Reliability, 2004, 14(2): 105-156.

[4] MCMINN P. Search-based software testing: Past, present and future[A]. Proceedings of IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops[C]. USA: IEEE, 2011. 153-163.

[5] HARMAN M, MANSOURI S A, ZHANG Y. Search-based software engineering: Trends, techniques and applications[J]. ACM Computing Surveys (CSUR), 2012, 45(1): Article No. 11. doi: 10.1145/2379776.2379787.

[6] 中国计算机学会. CCF 2013-2014 中国计算机科学技术发展报告[M]. 北京:机械工业出版社,2014.

[7] XANTHAKIS S, ELLIS C, SKOURLAS C, et al. Application of genetic algorithms to software testing[A]. Proceedings of the 5th International Conference on Software Engineering and its Applications, Toulouse, France[C]. USA: IGI Global, 1992. 625-636.

[8] PARGAS R, HARROLD M, PECK R. Test-data generation using genetic algorithms[J]. Software Testing Verification and Reliability, 1999, 9(4): 263-282.

[9] YAO Xiang-juan, GONG Dun-wei. Genetic algorithm-based test data generation for multiple paths via individual sharing[J]. Computational Intelligence and Neuroscience, 2014, 38(6): 1-12.

[10] YAO Xiang-juan, GONG Dun-wei, WANG Wen-liang. Test data generation for multiple paths based on local evolution[J]. Chinese Journal of Electronics, 2015, 24(1):46 – 51.

[11] 巩敦卫,张岩. 一种新的多路径覆盖测试数据进化生成方法[J]. 电子学报,2010,38(6):1299 – 1304.
GONG Dun-wei, ZHANG Yan. Novel evolutionary generation approach to test data for multiple paths coverage [J]. Acta Electronica Sinica, 2010, 38(6):1299 – 1304. (in Chinese)

[12] 张岩,巩敦卫. 基于搜索空间自动缩减的路径覆盖测试数据进化生成[J]. 电子学报, 2012, 40(5): 1011 – 1016.
ZHANG Yan, GONG Dun-wei. Evolutionary generation of test data for path coverage based on automatic reduction of search space[J]. Acta Electronica Sinica, 2012, 40(5):1011 – 1016. (in Chinese)

[13] TIAN Tian, GONG Dun-wei. Evolutionary generation approach of test data for multiple paths coverage of message-passing parallel programs[J]. Chinese Journal of Electronics, 2014, 23(2):291 – 296.

[14] ALETI Aldeida, GRUNSKE Lars. Test data generation with a Kalman filter-based adaptive genetic algorithm [J]. Journal of Systems and Software, 2015, 103(5):343 – 352.

[15] GUPTA M. Effective test data generation using genetic algorithms[J]. Journal of Engineering Computers & Applied Sciences, 2012, 1(2):17 – 21.

[16] EL-SERAFY A, EL-SAYED G, SALAMA C, et al. Enhanced genetic algorithm for MC/DC test data generation [A]. Proceedings of International Symposium on Innovations in Intelligent Systems and Applications[C]. USA: IEEE, 2015. 1 – 8.

[17] WINDISCH A, WAPPLER S, WEGENER J. Applying particle swarm optimization to software testing[A]. Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2007)[C]. Berlin: Springer, 2007. 1121 – 1128.

[18] 毛澄映,喻新欣,薛云志. 基于粒子群优化的测试数据生成及其实证分析[J]. 计算机研究与发展, 2014, 51(4):824 – 837.
MAO Cheng-ying, YU Xin-xin, XUE Yun-zhi. Algorithm design and empirical analysis for particle swarm optimization-based test data generation[J]. Journal of Computer Research and Development, 2014, 51(4):824 – 837. (in Chinese)

[19] MAO C. Generating test data for software structural testing based on particle swarm optimization[J]. Arabian Journal Forence & Engineering, 2014, 39(6): 4593 – 4607.

[20] JIANG S, SHI J, ZHANG Y, et al. Automatic test data generation based on reduced adaptive particle swarm optimization algorithm[J]. Neurocomputing, 2015, 158: 109 – 116.

[21] JIANG S, YI D, JU X, et al. An approach for test data generation using program slicing and particle swarm optimization[J]. Neural Computing and Applications, 2014, 25(7 – 8):2047 – 2055.

[22] 王令赛,姜淑娟,张艳梅,于巧. 基于正交搜索的粒子群优化测试用例生成方法[J]. 电子学报,2014,42(12): 2345 – 2351.
WANG Ling-sai, JIANG Shu-juan, ZHANG Yan-mei, YU Qiao. Test case generation based on orthogonal exploration and particle swarm optimization[J]. Acta Electronica Sinica, 2014, 42(12):2345 – 2351. (in Chinese)

[23] 姜淑娟,王令赛,薛猛,张艳梅,于巧,姚慧冉. 基于模式组合的粒子群优化测试用例生成方法[J]. 软件学报, 2016, 27(4):785 – 801.
JIANG S J, WANG L S, XUE M, et al. Test case generation based on combination of schema using particle swarm optimization[J]. Journal of Software, 2016, 27(4):785 – 801. (in Chinese)

[24] MONSON Chrstopher K, SEPPI Keven D. The Kalman swarm: a new approach to particle swarm motion in swarm optimization[A]. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)[C]. Berlin: Springer, 2004, Vol. 3102. 140 – 150.

[25] KENNEDY J. EBERHART R C. Particle swarm optimization[A]. Proceedings of IEEE International Conference on Neural Networks IV[C]. Piscataway: IEEE, 1995. 1942 – 1948.

[26] SHI Y, EBERHART R. A modified particle swarm optimizer[A]. Proceedings of IEEE International Conference on Evolutionary Computation[C]. USA: IEEE World Congress on Computational Intelligence, 1998. 69 – 73.

[27] KALMAN R E. A new approach to linear filtering and prediction problems[J]. Transactions of the ASME-Journal of Basic Engineering, 1960, 82: 35 – 45.

[28] CLERC Maurice. Particle Swarm Optimization[M]. USA: ISTE (International Scientific and Technical Encyclopedia), 2006.

[29] KENNEDY J, MENDES R. Population structure and particle swarm performance[A]. Proceedings of the Congress on Evolutionary Computation (CEC 02)[C]. USA: IEEE Computer Society Washington, 2002. 1671 – 1676.

[30] SUGANTHAN P N. Particle swarm optimiser with neighbourhood operator[A]. Proceedings of the Congress on Evolutionary Computation (CEC'99)[C]. USA: IEEE

Computer Society Washington,1999. 1958 – 1962.

[31] KRINK T, VESTERSTROM J S, Riget J. Particle swam optimization with spatial particle extension[A]. Proceedings of the Congress on Evolutionary Computation (CEC´02)[C]. USA: IEEE Computer Society Washington, 2002. 1474 – 1479.

[32] 胡旺,李志蜀. 一种更简化而高效的粒子群优化算法[J]. 软件学报,2007,18(4):861 – 868.
HU Wang, LI Zhi-Shu. A simpler and more effective particle swarm optimization algorithm[J]. Journal of Software, 2007, 18(4): 861 – 868. (in Chinese)

[33] JavaNCSS[OL]. http://www. kclee. de/clemens/java/javancss/,2017 – 02 – 20.

[34] MCMINN P. An identification of program factors that impact crossover performance in evolutionary test input generation for the branch coverage of C programs[J]. Information and Software Technology, 2013, 55(1): 153 – 172.

[35] XIAO M, EL-ATTAR M, Reformat M, et al. Empirical evaluation of optimization algorithms when used in goal-oriented automated test data generation techniques[J]. Empirical Software Engineering, 2007, 12(2): 183 – 239.

[36] KIFETEW F M, PANICHELLA A, LUCIA A D, et al. Orthogonal exploration of the search space in evolutionary test case generation[A]. Proceedings of International Symposium in Software Testing and Analysis[C]. USA: ACM, 2013. 257 – 267.

[37] KOREL B. Automated software test data generation[J]. IEEE Transactions on Software Engineering, 1990, 16(8): 870 – 879.

[38] 迟玉红,孙富春,王维军,喻春明. 基于空间缩放和吸引子的粒子群优化算法[J]. 计算机学报,2011,34(1):115 – 130.
CHI Yu-Hong, SUN Fu-Chun, WANG Wei-Jun, YU Chun-Ming. An improved particle swarm optimization algorithm with search space zoomed factor and attractor[J]. Chinese Journal of Computers, 2011, 34(1): 115 – 130. (in Chinese)

作者简介

薛　猛　男. 1979 年 4 月出生,江苏邳州人. 博士研究生,讲师,CCF 会员,主要研究领域为软件测试、测试数据生成.
E-mail: mgxue@ cumt. edu. cn

姜淑娟(通信作者)　女. 1966 年出生,山东莱阳人. 博士,教授,博士生导师,CCF 会员,主要研究领域为软件分析与测试、编译技术.
E-mail: shjjiang@ cumt. edu. cn