基于正交搜索的粒子群优化测试用例生成方法

王令赛,姜淑娟,张艳梅,于 巧
(中国矿业大学计算机科学与技术学院,江苏徐州 221116)

摘 要: 针对粒子群优化算法易出现早熟收敛的问题,本文提出一种基于正交搜索的粒子群优化测试用例生成方法.首先,利用奇异值分解来预测种群的进化方向,在其正交方向进行搜索,可避免已搜索过的区域,有助于跳出局部最优;然后,对粒子速度项进行改进,使其与正交方向保持一致,保证种群可持续受到正交方向的影响,有利于减少奇异值分解次数,降低时间消耗;最后,对每代最优个体进行局部搜索,以增强算法局部搜索能力.实验证明,本文方法在覆盖率、运行时间、进化代数等指标上均有优势.

关键词: 测试用例生成; 粒子群优化算法; 局部搜索; 奇异值分解
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2014)12-2345-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.12.002

Test Case Generation Based on Orthogonal Exploration and Particle Swarm Optimization

WANG Ling-sai, JIANG Shu-juan, ZHANG Yan-mei, YU Qiao
(*School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangsu* 221116, *China*)

Abstract: To solve the problem of premature convergence, this paper presents a method of generating test cases based on orthogonal exploration and particle swarm optimization. First, singular value decomposition is used to estimate the evolution direction and drives the population towards orthogonal direction, so that our method can avoid searching those traversed areas so far and jump out of local optimum. Then, we change the velocity so that it is able to be consistent with the orthogonal direction, and as a result, the population can be affected continually, which can decrease the frequency of singular value decomposition and reduce the time consumption. Finally, the local search is used for the best particle in each generation. The experimental results show that our method has advantages in coverage, running time, and the number of generations.

Key words: test case generation; particle swarm optimization; local search; singular value decomposition

1 引言

软件测试的目的是发现尽可能多的错误,传统测试利用手工输入测试用例的方式来引发错误,不仅耗时耗力,而且可能存在测试用例遗漏的问题.因此测试用例自动生成技术变得至关重要[1].基于搜索的测试用例生成技术[2]可以利用适应度函数,将测试用例生成问题转化成函数优化问题,进而利用一些人工智能搜索算法寻找最优解,最终达到程序所要求的覆盖标准.

人工智能搜索算法种类繁多,其中粒子群优化算法(Particle Swarm Optimization, PSO)由于简单高效,在软件测试领域已显现出较大优势.但是由于 PSO 算法本身容易出现早熟收敛的特点,Nie[3]、Zhu[4]以及史娇娇等人[5]均对算法的速度更新公式进行了一定的改进,旨在增加其随机搜索性能,跳出局部最优解,但是无法确定该改进方法是否会产生重复解.De Lucia[6,7]等人在遗传算法中使用了奇异值分解方法预测种群的进化方向,仅在进化方向的正交方向(可理解为垂直方向)生成新的解,避免了已经搜索过的区域,改善了种群多样性,有利于算法跳出局部最优,向全局最优方向进化.然而,由于新生成解的适应度值不一定优于其他个体,故在下一代进行选择操作时,新生成的解有可能被淘汰,因此为了维持种群多样性,必须频繁使用奇异值分解方法,但是这会产生大量的时间消耗,并且该方法主要依靠种群多样性的增加来增强算法的全局搜索能力,没有兼顾到局部搜索精度的提高.

收稿日期:2014-05-02;修回日期:2014-06-26;责任编辑:覃怀银
基金项目:国家自然科学基金(No.60970032);江苏省"333"工程;江苏省"青蓝工程";中央高校基本科研业务费专项资金资助(No.2013NB17)

鉴于此,本文提出一种基于正交搜索的 PSO 测试用例生成方法,将奇异值分解方法引入到 PSO 算法.首先每隔若干代对进化方向进行预测,在其正交方向生成新的解;然后对新解的速度项进行改进,使其与正交方向保持一致;最后,对每代的最优个体均利用交替变量法进行局部搜索.该方法既可以在正交方向探索到新的搜索区域,增强算法的全局搜索能力,又因速度项可持续受到正交方向的影响而减少奇异值分解次数,降低时间消耗.由于使用了局部搜索策略而增强了算法的局部搜索能力,使得算法的全局搜索能力和局部搜索能力相协调,从而提高测试用例生成效率.

2 基本概念

2.1 标准 PSO 算法

在 PSO 算法中[5],设种群含有的粒子数目为 N,搜索空间的维度为 D,第 i 个粒子的位置 $x_i=(x_{i1},x_{i2},\cdots,x_{iD})$,速度 $v_i=(v_{i1},v_{i2},\cdots,v_{iD})$,粒子历史上搜到的最优位置为 $p_i=(p_{i1},p_{i2},\cdots,p_{iD})$,即局部最优解.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下:

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

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

其中学习因子 c_1 和 c_2 均为常数,r_1 和 r_2 均是 0~1 之间的随机数,惯性权重 w 从 0.9 到 0.4 线性递减.

2.2 奇异值分解

奇异值分解(Singular Value Decomposition,SVD)一种矩阵分解方法,在信号处理、统计学等方面有广泛应用,也常用来对气候或者降水情况进行预测.假设将第 t 代种群 P_t 中粒子的位置 $\boldsymbol{X}_t$ 表示为矩阵形式:

$$\boldsymbol{X}_t=\begin{pmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & x_{ij} & \vdots \\ x_{N1} & \cdots & x_{ND} \end{pmatrix}$$

其中 N 表示种群含有的粒子数目,D 表示每个粒子搜索空间的维度.x_{ij}表示第 i 个粒子在第 j 维空间上的位置.使用奇异值分解之后,第 t 代种群 P_t 的位置可表示为[6]:$\boldsymbol{X}_t=\boldsymbol{U}_t\cdot\boldsymbol{\Sigma}_t\cdot\boldsymbol{V}_t^{\mathrm{T}}$.其中 $\boldsymbol{\Sigma}_t$ 是一个对角矩阵,即除了主对角线之外的元素皆为 0,主对角线上的元素 σ 满足 $\sigma_0\geqslant\sigma_1\geqslant\cdots\geqslant\sigma_{\min(N,D)}$.$\boldsymbol{V}_t$ 的每一维的列向量均表示种群分布的方向,σ_i 表示 $\boldsymbol{V}_t$ 所对应的第 i 维列向量的重要程度.

定义 1 种群进化方向[6]:种群内个体的位置随着进化代数 t 而改变的方向,是一个 D 维向量.种群进化方向针对的是进化过程中种群这个整体,并非针对个体.

定义 2 种群正交方向[6]:与进化方向向量内积为 0.在几何意义上,正交方向可以理解为与进化方向相垂直的方向.

假设每隔 k 代对种群进行一次奇异值分解,则两个种群 P_t 和 P_{t+k}粒子位置的分布情况如图 1 所示,经过 k 代之后,种群由 P_t 较为分散的状态变为 P_{t+k}较为收敛的状态,其中 V_{evol}为种群的进化方向,V_{orth}为正交方向.

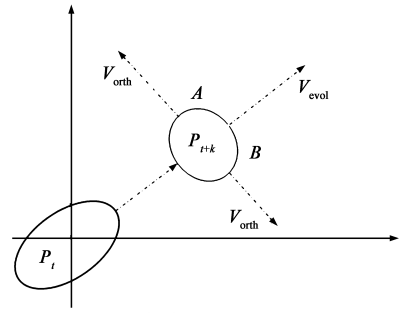


图1 种群进化方向预测图

3 基于正交搜索的 PSO 测试用例生成

3.1 测试用例生成模型

基于正交搜索的 PSO 测试用例生成模型如图 2 所示,包含测试环境构造、测试运行和算法 3 个模块:

(1)测试环境构造模块完成如下工作:对源程序进行静态分析、选择目标路径并对其进行插桩、提取有用的参数作为算法模块的输入.

(2)测试运行模块利用算法模块传递进来的测试用例驱动待测程序运行,进而计算出适应度值返回给算法模块.本文采用分支距离法[5]构造适应度函数.

(3)算法模块是本文的核心部分,本文对原始 PSO 算法的改进分为两部分:①正交搜索部分:对粒子的位置进行了奇异值分解,在正交方向上生成了新的粒子,进而探索到新的区域,增强了算法的全局搜索性能.②局部搜索部分:本文使用了交替变量法对每代最优解进行局部搜索,并且对其局部搜索范围进行了限制,一定程度上提高了算法的局部搜索精度,还可以使算法的全局搜索能力和局部搜索能力相协调.

3.2 基于正交搜索的 PSO 方法

在 PSO 算法进化的过程中,为防止种群陷入局部最优,本文每隔 k 代对种群进行一次奇异值分解,则 $\boldsymbol{X}_t=\boldsymbol{U}_t\cdot\boldsymbol{\Sigma}_t\cdot\boldsymbol{V}_t^{\mathrm{T}}$,$\boldsymbol{X}_{t+k}=\boldsymbol{U}_{t+k}\cdot\boldsymbol{\Sigma}_{t+k}\cdot\boldsymbol{V}_{t+k}^{\mathrm{T}}$.进化方向矩阵 $\boldsymbol{V}=\boldsymbol{V}_{t+k}-\boldsymbol{V}_t$,其中,$\boldsymbol{V}$ 中的每一维列向量均表示种群的进化方向,其重要程度与对角矩阵 $\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_{t+k}-\boldsymbol{\Sigma}_t$ 中的对角线元素 σ 相对应.各个粒子的位置称为正交位置矩阵 $\mathrm{Orth}(\boldsymbol{X}_{t+k})$,其计算方法如式(3)所示:

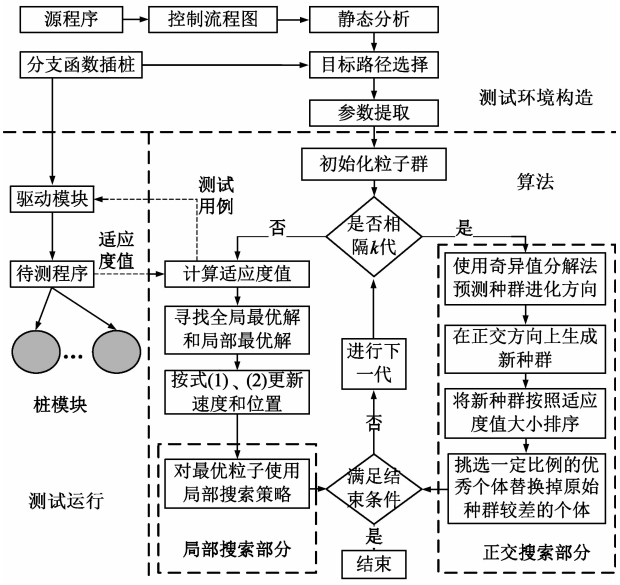


图2　基于正交搜索的PSO测试用例生成模型

$$\mathrm{Orth}(\boldsymbol{X}_{t+k}) = \boldsymbol{U}_{t+k}\cdot(\boldsymbol{\Sigma}_{t+k}+\boldsymbol{\Sigma})\cdot(\boldsymbol{V}_{t+k}+\mathrm{Orth}(\boldsymbol{V}))^{\mathrm{T}} \tag{3}$$

$\mathrm{Orth}(\boldsymbol{X}_{t+k})$中的元素 x_{ij}表示第 i 个粒子的第 j 维的位置,若通过式(3)的计算,其值超过粒子位置规定的范围,则将其设为规定范围内的一个随机值.其中 $\mathrm{Orth}(\boldsymbol{V})$称为正交方向矩阵,其列向量均与进化方向矩阵 $\boldsymbol{V}$ 中的列向量正交.由于每个向量的正交向量并不唯一,本文的计算方法如下:①先判断矩阵 $\boldsymbol{V}$ 中的每个列向量含有的元素数目是奇数还是偶数.②若为偶数,则每两个相邻元素作为一对,将它们互相交换,并对前一个元素乘以 -1.③若为奇数,则将任意一位元素置零,剩下的元素按偶数的方法来计算.

正交种群中每个粒子的速度计算步骤如下:

Step1　在 $\boldsymbol{\Sigma}$ 的对角线上寻找最大的元素,将其所对应的行号赋值给 σ_best,即表示正交方向矩阵 $\mathrm{Orth}(\boldsymbol{V})$中第 σ_best 列列向量的重要程度最高,最能反映出种群的正交方向.

Step2　将 $\mathrm{Orth}(\boldsymbol{V})$中第 σ_best 列列向量的各个元素值均赋值给 $v_\mathrm{best}[\,]$,由于列向量均是单位向量,所以将 $v_\mathrm{best}[\,]$扩大 $v_\max$ 倍,$v_\max$ 为速度最大值.

Step3　将 $v_\mathrm{best}[\,]$赋值给正交种群中每个粒子的速度项.

将正交种群按照适应度值大小排序,挑选若干优秀个体替换掉原始种群中最差的同数目的个体,挑选比例为 select_rate,范围在 0~1 之间.

将正交种群中的每个粒子经历过的最好位置设为粒子本身,即 $p_i = x_i$,由式(1)可知,$c_1 r_1(p_i - x_i) = 0$,若新生成的粒子继续进化,则下一代速度 v_i^{t+1} 受两个方面的影响:①当前速度 v_i^t,由上述步骤可知是最重要的正交方向向量扩大若干倍得到的,所以 v_i^{t+1} 受正交方向的影响.②与全局最优解的差值 $g^t - x_i^t$,即向全局最优解的方向靠近.因此随着种群继续进化,由于粒子在进化方向和正交方向均有速度分量,粒子将会避免已经搜索过的区域,向从未搜索到的区域 A 和 B 区进化,如图 1 所示.

对于遗传算法来说,运用奇异值分解生成的新个体若是适应度值较差,则下一代进行选择操作时,这些个体就可能被淘汰掉,而无法对种群进化产生来自正交方向的影响,因此对于遗传算法来说必须频繁地对种群进行奇异值分解,以保证正交方向的个体存在于种群中,进而维持种群多样性.

而本文方法即使若干代之内不再进行奇异值分解,粒子仍然拥有正交方向的分量,指导着粒子继续在正交方向进化.因此,基于正交搜索的 PSO 算法生成的新解不仅探索了新的区域,而且在奇异值分解次数少的情况下,依然可以在正交方向进行搜索,降低了时间消耗.

3.3　局部搜索策略

本文中选择的局部搜索策略是交替变量法(Alternating Variable Method, AVM)[8],其主要思想是轮流调整每个输入变量的值,如果适应度值不能再提高,则接着修改下一个变量的值,直到调整完所有的变量值都无法提高适应度值为止.

本文对原始的 AVM 进行了如下改进:①对每个变量搜索范围限制为 L,可防止因局部搜索次数过多而造成的效率降低,协调算法的全局和局部搜索能力.②由于 AVM 搜索的效率与搜索的起始位置有直接关系,若起始位置较差,则需要更多的搜索次数.故本文只对每代最优粒子使用 AVM,若搜索后适应度值无提升,则该粒子保持原位置不变.具体算法如算法 1 所示:

算法 1　局部搜索策略

```
Input: the positon x[] of the best particle in P_t
Output: the positon x[] of the best particle in P_t
1 begin
2 for each dimension i in D do
3     if f(x[i]-1) < f(x[i]) and f(x[i]+1) < f(x[i]) then goto
      next dimension
4     if f(x[i]-1) > f(x[i]+1) then h = -1 else h = 1
5     while f(x[i]+h) > f(x[i]) and |h| ≤ L
6       do x[i] = x[i] + h  h = 2h
7       let l = min(x[i]-h/2, x[i]+h), r = max(x[i]-h/2, x[i]
        +h)
8     while l ≠ r do
```

```
9       if f(⌊(l + r)/2⌋) > f(⌊(l + r)/2⌋ + 1) then r = ⌊(l + r)/2⌋
        else l = ⌊(l + r)/2⌋ + 1
10      end_while
11      x[i] = l
12    end_while
13 end_for
14 end_Algorithm1
```

该算法本质上是一个二分搜索法,第 2 行对最优个体位置的每一维度 i 进行搜索;第 3 行对位置 $x[i]$进行改变,若适应度值均没有提高则进行下一维的搜索;第 4 行,确定搜索方向;第 5 ~ 12 行,若适应度值一直提高,且搜索范围小于 L,则继续搜索,若适应度值减小,则反向搜索.第 13 行,对下一维 $i+1$ 进行搜索,跳转到第 2 行继续进行.

4 实验

4.1 实验对象

实验选取了 7 个测试程序进行测试,其详细信息如表 1 所示.

表 1 测试程序

程序编号	程序名称	变量个数	搜索范围	程序来源
1	Equals	20	$[-10^2,10^2]$	Java Collections
2	Transform	10	[0,255]	文献[9]
3	luhnCheck	16	$[-10^3,10^3]$	Apache Commons Valid
4	Quadratic	3	$[-10^4,10^4]$	文献[7]
5	NextDate	3	$[-10^4,10^4]$	文献[7]
6	Triangle	3	$[-10^4,10^4]$	文献[9]
7	density	4	$[-10^4,10^4]$	Apache Commons Math

程序 1 的目标路径是两个数组是否相等,本实验将两个数组的长度均设为 10.程序 2 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.程序 3 目的是检测银行卡或者信用卡号码是否有效,本实验将号码长度固定为 16,若每个号码均在 0 ~ 9 之间,且满足 luhnCheck 函数的条件,即为目标路径.程序 4 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 5 的目标路径是年月日满足 1912 ~ 2050 年间闰年 2 月 29 日.程序 6 的目标路径为等边三角形.程序 7 是三角形分布的概率密度函数,目标路径是某一随机变量 x 等于中位数 c.

PSO 代表原始粒子群优化算法,O-PSO 表示本文中只使用了正交搜索的 PSO 方法,OL-PSO 表示应用了正交搜索和局部搜索策略相结合的方法,APSO 表示文献[4]的方法,四种方法的实验参数设置如下:种群数目 30,最大进化代数 1000,学习因子 $c_1=c_2=2$,惯性权重 w 范围(0.4,0.9].速度最大值为 $x_max/10$,其中 x_max为搜索范围最大值.O-PSO 和 OL-PSO 中 k 为两次奇异值分解之间相隔的进化代数,设为 50.Select_rate 为新生成的正交种群可以进入原始种群的比例,设为 0.2.OL-PSO 中 L 为局部搜索范围,设为 8.APSO 中 tz 为局部收敛次数,设为 10.

4.2 实验设计

为了对本文方法进行更深入的研究,实验设计了以下几个问题:

问题 1 O-PSO 方法是否能提高测试用例生成的效率?

O-PSO 方法是在正交方向对空间进行搜索,其找到的解不一定比当前解的适应度值好,但是却可以在种群陷入早熟收敛的时候,通过引入正交方向的解,来帮助种群跳出局部最优解,从而向全局最优解进化.为了研究 O-PSO 是否对测试用例生成过程有效,对哪一类程序有效,我们设计了实验 4.3.1.

问题 2 O-PSO 方法的参数对算法有何影响?

O-PSO 方法的参数主要有两个,两次奇异值分解之间相隔的进化代数 k 和从正交种群中挑选出的优秀个体的比例 select_rate.k 的值越小,说明奇异值分解越频繁,select_rate 越大,说明从正交种群被挑选的个体越多.为了确定合理的参数值,探索该参数对算法性能的影响,我们设计了实验 4.3.2.

问题 3 OL-PSO 方法相比较于其他文献的方法是否有优势?

OL-PSO 方法即将正交搜索和局部搜索策略相结合的方法,由于正交搜索的奇异值分解耗时较多,而每代均运用局部搜索策略更有可能引入额外的时间消耗,那么该方法相较于其他文献的方法来说,覆盖率和时间方面是否还均存在优势,为此我们设计了实验 4.3.3 和 4.3.4.

4.3 实验结果

4.3.1 O-PSO 与 PSO 算法的比较

表 2 分别为 O-PSO 和 PSO 算法对测试程序执行 100 次后的平均覆盖率和进化代数.覆盖率表示在最大进化代数 T 之内覆盖到目标路径的次数除以 100,进化代数表示覆盖到目标路径所需要的进化代数的平均值,若 T 代之内无法覆盖目标路径,则进化代数为 T.

由表 2 可以看出,O-PSO 在平均覆盖率和进化代数上都比 PSO 有了一定程度的提高.程序 1、程序 2 和程序 3 均带有循环且参数个数较多而较难覆盖,PSO 覆盖率较低,而用正交搜索方法改进的 O-PSO 覆盖率均达到了 100%,程序 3 中,O-PSO 比 PSO 的覆盖率提高了 58%,是改进效果最好的程序.程序 4 取值范围较大且逻辑较复

杂,其每个输入变量均需在[-10^4, 10^4]范围内满足 $b^2-4ac=0$ 且 $a\neq 0$, $c\neq 0$ 的条件,故其目标路径较难覆盖,O-PSO 的覆盖率只提高了 4%,并且也未达到 100% 覆盖.而对于程序 5、程序 6 和程序 7,PSO 均达到了覆盖率 100%,O-PSO 在进化代数上没有明显的提高,主要是由于这三个程序相对较简单,由于 O-PSO 该方法主要增加了种群的多样性,算法的局部搜索能力并未提高,因此针对这三个程序 O-PSO 效果并不明显.

表 2　O-PSO 和 PSO 实验结果对比

测试程序	O-PSO		PSO	
	覆盖率	进化代数	覆盖率	进化代数
1	100%	216.51	80%	334.73
2	100%	359.22	59%	613.3
3	100%	485.29	42%	766.6
4	52%	718.36	48%	762.62
5	100%	250.38	100%	275.51
6	100%	307.33	100%	388.73
7	100%	62.14	100%	63.5

4.3.2　O-PSO 中的参数对实验结果的影响

在 O-PSO 算法中,每相隔 k 代进行一次正交搜索,将参数 k 的范围以 10 为步长进行变化.图 3 为 7 个程序分别运行程序 100 次,不同 select_rate 取值下随着参数 k 取值变化的趋势图.其中横坐标轴表示 k 的取值,纵坐标轴表示进化代数.

由图 3 可知,k 值较小时,不同 select_rate 下的进化代数差异较大,而随着 k 值的增加和 select_rate 的变化,每个程序的进化代数最终都稳定在一个区间内变化,到 $k=100$ 时,进化代数有稍微上升的趋势.

从 select_rate 的取值情况来看,取值越大,平均进化代数越大,这种情况在 $k=1$ 时效果更加明显.select_rate 代表着从正交种群中挑选进入粒子群的个体比例,由于正交种群是为了改善种群多样性,因此不一定都有益于种群进化,若是 select_rate 值较高,那么种群中大部分都是经过正交搜索之后的个体,则会干扰正常的进化过程,并且 select_rate 较高也增加了个体的替换操作,增加时间开销.因此,从图 3 可知,select_rate 取值 0.2 较为合理.

从 k 的取值情况来看,$k=1$ 时,大多数程序的效果较好,均优于或者相当于 k 增加之后的效果.但是对于程序 1 来说,其目标路径是两个数组元素相等,需要局部搜索精度较高的算法,而每代进行一次正交搜索不能增加算法的局部搜索精度,故 $k=1$ 时效果反而不好.由于 $k=1$ 时需要消耗大量的运行时间,并且其对于算法的提高并不是特别明显,因此本文将 k 设为 50,此时各个程序的进化代数均趋于稳定.

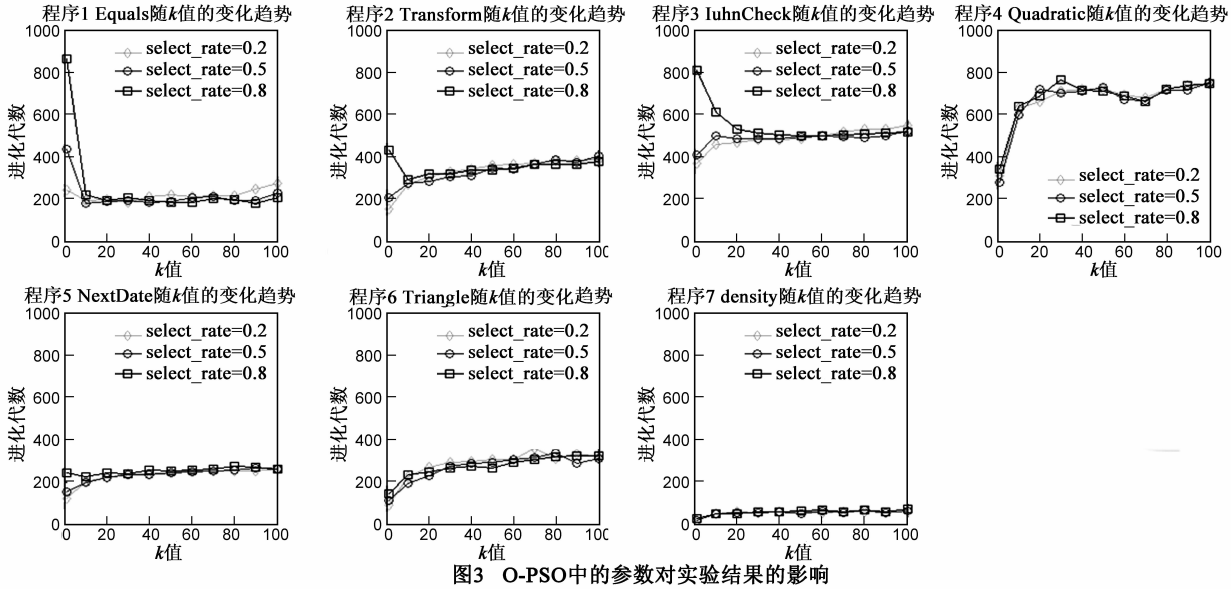


图3　O-PSO中的参数对实验结果的影响

4.3.3　OL-PSO 与 APSO 方法的比较

因为 O-PSO 方法主要增强了算法的全局搜索能力,算法的局部搜索能力并未提高,为了同时提高算法的局部搜索精度,本文使用了正交搜索和局部搜索策略相结合的方法 OL-PSO.表 3 为 OL-PSO 和文献[4]中的 APSO 均对测试程序执行 100 次后的平均进化代数、覆盖率和运行时间的结果.

由表 3 可以看出,本文的 OL-PSO 方法相对于参考文献[4]的 APSO 方法来说,大部分程序在覆盖率和进化代数均有所提高.但是对于程序 4 来说,OL-PSO 的覆盖率(28%)不仅不如 APSO(46%),而且还低于 O-PSO(52%)和原始 PSO(48%).这是由于该程序的目标路径 $b^2-4ac=0$

并非单峰值函数,因此使用 AVM 方法对变量进行交替搜索没有效果,反而会因为搜索到次优解而使得种群多样性降低,从而导致该程序总的覆盖率降低,不如单纯使用正交搜索的 O-PSO 方法和原始方法好.

表 3 OL-PSO 和 APSO 的实验结果对比

测试程序	OL-PSO			APSO		
	覆盖率	进化代数	时间(s)	覆盖率	进化代数	时间(s)
1	100%	3.16	0.109	100%	194.76	1.357
2	100%	126.14	1.809	63%	475.56	1.965
3	100%	276.01	11.671	69%	690.09	4.947
4	28%	834.84	1.855	46%	669.23	0.952
5	100%	136.97	0.697	100%	152.89	0.382
6	100%	67.71	0.333	100%	150.41	0.281
7	100%	7.88	0.092	100%	30.22	0.125

对于程序 1、程序 2 和程序 7,OL-PSO 在覆盖率、进化代数和运行时间三个指标上均优于 APSO,但是对于程序 1 和程序 7 来说,OL-PSO 中的局部搜索策略占主导作用,由于进化代数过小,而使得正交搜索机制没有发挥其效果.

对于程序 3、程序 5 和程序 6,OL-PSO 在覆盖率和进化代数上均优于 APSO,但是在运行时间上比 APSO 稍长.程序 3 虽然覆盖率提高的最明显,但是相应的运行时间也最长,主要是因为其变量数目较多,运用 AVM 对每个变量轮流搜索较为耗时.对于程序 5 和程序 6 来说,虽然是靠时间代价换取进化代数的降低,但是如果对局部搜索范围 L 加以提高,则其运行时间可以低于 APSO.

4.3.4 OL-PSO 和 APSO 方法进一步的比较

在 4.3.3 节的实验中,在程序 5 和程序 6 的运行时间指标上,APSO 优于本文方法,主要是因为本文对 AVM 的搜索范围 L 进行了限制.为了进一步比较本文方法与 APSO 方法,本节将对 L 进行变化,图 4 为两方法在进化代数和运行时间上随着 L 的变化取值,其中 L 的取值范围为 $2^0,2^1,\cdots,2^4$.

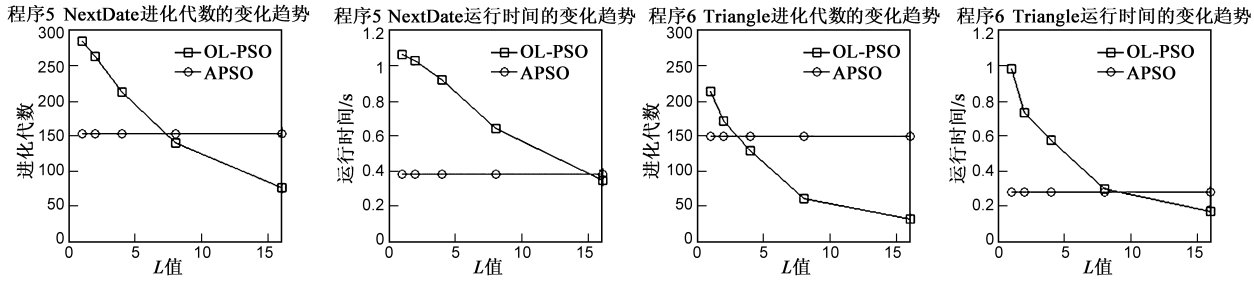


图4 OL-PSO和APSO方法进化代数和运行时间对比图

如图 4 所示,随着局部搜索范围 L 的增加,OL-PSO 的局部搜索能力逐渐增强,程序 5 在 $L=8$ 时进化代数低于 APSO,在 $L=16$ 时,运行时间低于 APSO.程序 6 在 $L=4$ 时进化代数低于 APSO,在 $L=16$ 时,运行时间低于了 APSO.如果 L 继续增加,则 OL-PSO 算法的局部搜索能力表现的会更加明显,所以对于程序 5 和程序 6 在搜索范围 L 逐渐增大($L\geqslant 16$)时,OL-PSO 在进化代数和运行时间两个指标上均优于 APSO.

4.4 实验分析

通过 4.3 部分的实验结果,我们对 4.2 实验设计部分提出的 3 个问题有如下回答:

回答问题 1 O-PSO 方法是否能提高测试用例生成的效率?

O-PSO 方法针对参数个数较多且较难覆盖的程序来说,覆盖率有明显提高,主要是因为该方法搜索到了新的区域,增强了种群多样性,使其跳出了局部最优解,向全局最优解进化.对于参数少且逻辑简单的程序,原始 PSO 方法就可以达到覆盖率 100%,而 O-PSO 方法只增强了全局搜索能力,并未兼顾到局部搜索能力,所以测试用例生成效率并无明显提高.

回答问题 2 O-PSO 方法的参数对算法有何影响?

k 值越小则奇异值分解越频繁,时间消耗越大,而且 k 值为 1 时与稳定之后的进化代数相差并不大,因此为了平衡算法性能和时间消耗这两个指标,将 k 值设为 50.Select_rate 较大时,则由于正交粒子数目过多而干扰到了算法的正常进化过程,使算法效率变差.故 select_rate 取值为 0.2 时较为合理,此时既可以适当的增加种群多样性,又不至于因为个体替换操作而增加太多的时间消耗.

回答问题 3 OL-PSO 方法相比较于其他文献的方法是否有优势?

相比于文献[4]的 APSO 方法,本文方法大部分测试程序在覆盖率方面均有所提高,而且运行时间上,OL-PSO 方法也低于 APSO.并且本文方法随着搜索范围 L 的增加,进化代数和运行时间都有所减少.因此本文的 OL-PSO 方法在覆盖率、进化代数和时间消耗等指标上均优于 APSO 方法.

5 相关工作

文献[3~5]均利用 PSO 解决测试用例生成问题,并且针对 PSO 算法易早熟收敛的特点对速度公式进行调节,或者改进惯性权重,旨在增加 PSO 的随机搜索性能,但是新生成的解质量如何,是否曾经被搜索过,无法确定.而本文中采用正交搜索的 PSO 方法,避免了搜索过的区域,探索了新的区域,有助于算法向全局最优解收敛.最后实验证明,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,而对于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,无法实现 100% 覆盖,本文方法则效果更佳.

文献[6,7]均将奇异值分解方法应用到遗传算法中,但由于遗传算法中存在选择操作,若是适应度值较差,则来自正交方向的个体可能被淘汰掉,而无法对种群进化产生来自正交方向的影响,因此对于遗传算法来说,必须频繁地对种群进行奇异值分解,以保证正交方向的个体存在于种群中,进而维持种群多样性.

而由于本文方法的速度项与正交方向保持一致,即使若干代之内不再进行奇异值分解,粒子仍然拥有正交方向的分量,指导着粒子继续在正交方向进化.并且本文方法还增加了局部搜索策略,对每代最优解均用 AVM 方法进行局部寻优,同时增强了算法的局部搜索能力.

6 结论及进一步工作

本文通过将基于正交搜索的 PSO 方法与局部搜索策略相结合,应用到测试用例生成中,对算法的全局搜索性能和局部搜索性能均有一定改善,最后实验部分通过 7 个测试程序对方法进行了验证,实验证明本文方法对测试用例生成效率有了一定的提高.

本文的测试程序用到了数组,但实验预先确定了数组的长度,而实际测试中,长度是不确定的.但不确定长度的数组在编程实现时有难度,比如 PSO 算法的更新公式建立在维度确定的基础上,如果粒子维度不能确定,那么下一代粒子的位置和速度如何生成也值得思考.如何测试不确定长度的数组类型是本文进一步研究的问题.

参考文献

[1] Bertolino A. Software testing research: Achievements, challenges, dreams[A]. Future of Software Engineering (FOSE '07)[C]. Minneapolis: IEEE, 2007. 85 – 103.

[2] Harman M, Kim S G, Lakhotia K, et al. Optimizing for the number of tests generated in search based test data generation with an application to the oracle cost problem[A]. Proceedings of the Third IEEE International Conference on Software Testing Verification and Validation -Workshops (ICSTW 2010)[C]. Paris: IEEE, 2010. 182 – 191.

[3] Nie P. A PSO Test case generation algorithm with enhanced exploration ability[J]. Journal of Computational Information Systems, 2012, 8(14): 5785 – 5793.

[4] Zhu X M, Yang X F. Software test data generation automatically based on improved adaptive particle swarm optimizer[A]. Proceedings 2010 International Conference on Computational and Information Sciences[C]. Chengdu: IEEE, 2010. 1300 – 1303.

[5] 史娇娇,姜淑娟,韩寒,王令赛. 自适应粒子群优化算法及其在测试数据生成中的应用研究[J]. 电子学报, 2013, 41(8): 1555 – 1559.
Shi Jiaojiao, Jiang Shujuan, Han Han, Wang Lingsai. Adaptive particle swarm optimization algorithm and its application in test data generation[J]. Acta Electronica Sinica, 2013, 41(8): 1555 – 1559. (in Chinese)

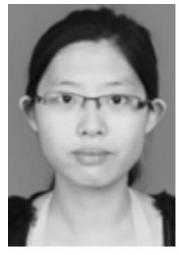
[6] De Lucia A, Di Penta M, Oliveto R, et al. Estimating the evolution direction of populations to improve genetic algorithms[A]. Proceedings of the 14th International Conference on Genetic and Evolutionary Computation[C]. Philadelphia: ACM, 2012. 617 – 624.

[7] Kifetew M F, Panichella A, De Lucia A, et al. Orthogonal exploration of the search space in evolutionary test case generation[A]. Proceedings of the 2013 International Symposium on Software Testing and Analysis[C]. Lugano: ACM, 2013. 257 – 267.

[8] Kempka J, McMinn P, Sudholt D. A theoretical runtime and empirical analysis of different alternating variable searches for search-based testing[A]. Proceeding of the 15th Annual Conference on Genetic and Evolutionary Computation Conference [C]. Amsterdam: ACM, 2013. 1445 – 1452.

[9] 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.

作者简介



王令赛　女,1989 年 3 月出生于河北清河.中国矿业大学研究生,主要研究领域为软件测试、测试数据生成.
E-mail: wls_007@126.com

姜淑娟(通信作者)　女,1966 年 12 月出生于山东莱阳.现为中国矿业大学计算机科学与技术学院教授、博士生导师,CCF 会员.主要研究领域为编译技术、软件工程等.
E-mail: shjjiang@cumt.edu.cn