基于自适应粒子群优化的组合测试用例生成方法

包晓安 杨亚娟 张 娜 林青霞 俞成海
(浙江理工大学信息学院 杭州 310018)

摘 要 最小覆盖表生成是组合测试研究的关键问题。基于演化搜索的粒子群算法在生成覆盖表时能得到较优的结果,但其性能受配置参数的影响。针对此问题,将 one-test-at-a-time 策略和自适应粒子群算法相结合,以种群粒子优劣为依据对惯性权重进行自适应调整,使其在覆盖表生成上具有更强的适用能力。为进一步提升算法性能,构造了一个优先级度量函数用于度量每个组合的权值,优先选取权值最高的组合用于单条测试用例的生成。最后,编程实现该算法,并将其与原有粒子群算法在组合测试用例集生成上展开对比性实验分析,结果证实该算法在规模和执行时间上具有竞争力。

关键词 组合测试,覆盖表生成,粒子群优化,自适应策略
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.06.030

Test Case Generation Method Based on Adaptive Particle Swarm Optimization

BAO Xiao-an YANG Ya-juan ZHANG Na LIN Qing-xia YU Cheng-hai
(School of Information Science and Technology, Zhejiang Sci-tech University, Hangzhou 310018, China)

Abstract Obtaining minimum coverage array is one of the key issues in the combination test. Particle swarm optimization(PSO), as one of the evolutionary search based methods, can obtain the smallest covering arrays, but its performance is significantly impacted by the parameters. To solve this problem, we combined one-test-at-a-time strategy and particle swarm optimization and proposed an adaptive particle swarm optimization algorithm. Based on the quality of the particles in the population, the strategy adaptively adjusts inertia weights which makes it have stronger ability of application. In order to further improve the performance of the algorithm, we constructed a priority measure function which is used to measure the weight of each combination, and we preferred to select a combination which has the highest weight to generate a single test case. Finally the paper implemented the algorithm by programming, and compared this approach with the original particle swarm optimization algorithm in test suite size and generation time. The results show the competitiveness of this approach.

Keywords Combination testing, Covering array generation, Particle swarm optimization, Adaptive algorithm

1 引言

软件测试是软件工程中保证软件质量的至关重要的环节。理想的软件测试方法需要同时具有高错误检测能力、低成本消耗和强适用性等特点。组合测试作为一种基于规约的软件测试方法,旨在从待测软件面临的庞大组合空间中选取少量但有效的测试用例,生成覆盖程度高、揭错能力强的测试用例集。实证研究表明[1]:组合测试技术可以用小规模的测试用例集完成高质量的软件测试。

近年来,元启发式搜索算法成为软件测试领域的一个活跃的分支[2],诸如模拟退火、遗传算法、蚁群算法等经典的元启发式搜索算法及其变体均被改造用于构造组合测试用例集[3]。粒子群算法作为一种较新的启发式搜索算法,具有易理解、易实现、全局搜索能力强等特点,目前也被应用于该领域。陈翔等人的研究表明:粒子群算法在覆盖表生成规模和执行时间上具有竞争力[4-6]。

目前,在利用粒子群优化生成覆盖表的相关研究中,主要实现了两两组合覆盖测试用例生成,并未考虑更高覆盖力度以及交互力度不统一的情况;同时,参数的选取对算法性能有很大影响,绝大多数研究工作通常选取一组经验参数,没有考虑到参数选择在不同覆盖表上的特殊性。针对此问题,聂长海等人[7]使用正交实验设计进行参数调优试验,结果表明:参数调优需要花费大量时间,且并不存在一组通用的参数配置对任意覆盖表都能生成最优结果。因此,在粒子群算法具有

到稿日期:2016-05-06 返稿日期:2016-09-06 本文受国家自然科学基金项目(61379036,61502430),浙江省自然科学基金项目(LY12F02041),浙江省重大科技专项重点工业项目(2014C01047),浙江理工大学521人才培养计划项目资助。

包晓安(1973—),男,硕士,教授,主要研究方向为自适应软件、软件测试与智能信息处理;**杨亚娟**(1992—),女,硕士,主要研究方向为软件工程、软件测试,E-mail:150807315@qq.com;**张 娜**(1977—),女,硕士,副教授,主要研究方向为软件工程、软件测试;**林青霞**(1994—),女,硕士,主要研究方向为软件测试、形式化方法;**俞成海**(1975—),男,硕士,副教授,主要研究方向为信息安全、软件架构与移动平台应用,E-mail:ych@zstu.edu.cn。

良好的鲁棒性的前提下，使用自适应策略对参数进行调整能使该算法在覆盖表生成上具有更强的适用能力。

基于上述考虑，在充分了解软件中因素间的交互作用的基础上，本文将 one-test-at-a-time 策略和自适应粒子群算法相结合，根据粒子的优劣对惯性权重进行自适应调整，提出了一种可处理任意覆盖强度的组合测试用例生成方法。实验结果表明，相对于原始粒子群算法，本文提出的算法在测试用例集规模和执行时间上均具有一定的优势。

2 研究背景

2.1 组合测试

假设一个待测软件系统(SUT)受到 n 个独立因素的影响，这些因素形成一个有限集合 $F=\{f_1,f_2,\cdots,f_k\}$，其中第 i 个因素 f_i 拥有 l_i 个可选取值，则其对应的有效取值集可表示为 $D_i=\{1,2,\cdots,l_i\}$。那么，可以称 n 元组 $test=(x_1,x_2,\cdots,x_n)(x_1\in D_1,x_2\in D_2,\cdots,x_n\in D_n)$ 为 SUT 的一条测试用例。

传统的组合设计方法都是基于覆盖矩阵的，即将组合测试用例集用一个矩阵来表示，每一行代表一个测试用例，每一列代表 SUT 的一个因素，每一项代表相应因素的取值。根据覆盖强度的不同，组合测试覆盖表可分为固定力度组合覆盖表 $CA(N;t,k,v)$ 和可变力度组合覆盖表 $VSCA(N;t,k,v,C)$ 两种类型。其中，$CA(N;t,k,v)$ 表示一个 $N\times k$ 的矩阵，每列可选取值的个数为 v，任意的 $N\times t$ 子矩阵包含了在 v 值域上的所有 t 元组；$VSCA(N;t,k,v,C)$ 在满足上述情况的基础上还包含一个或多个子矩阵 C，并且 C 的覆盖强度大于 t。

以一个网络软件系统为例(见表 1)，该系统受到 5 个因素的影响，可选取值的数量依次为 2,2,2,3,3，可表示为 $F=\{2^3\times 3^2\}$。若对该系统进行穷举测试，则需要 72 条测试用例，且随着因素个数和相应的可选取值个数的增加，穷举测试所需的代价将呈指数增加。Kuhn 等人[8]发现 70%的错误可通过检测任意两个参数之间的交互关系找出。因此，假设只考虑两个因素间的相互作用，则可以找到一个 $t=2$ 的固定力度覆盖表 $CA(9;2^3 3^2,2)$，如表 2 所列，只需 9 条测试用例即可覆盖任意两个因素间的所有组合。

表 1 网络软件系统的配置

OS	Web server	Web brower	Connection type	Database
Windows	iPlanet	Chrome	LAN	Oracle
Mac	Apache	IE	PPP ISDN	Access SQL

表 2 覆盖表 $CA(9;2^3 3^2,2)$

No.	OS	Web server	Web brower	Connection type	Database
1	Windows	Apache	IE	ISDN	Access
2	Windows	Apache	Chrome	LAN	SQL
3	Windows	Apache	Chrome	PPP	Oracle
4	Windows	iPlanet	IE	PPP	SQL
5	Mac	iPlanet	Chrome	PPP	Access
6	Mac	iPlanet	IE	LAN	Oracle
7	Mac	iPlanet	Chrome	ISDN	SQL
8	Mac	Apache	Chrome	LAN	Access
9	Mac	Apache	Chrome	ISDN	Oracle

但在实际的 SUT 中，任意因素间的相互作用并不一定是一致的，部分因素间的联系可能会更加紧密，为了对这些相互作用较强的因素进行更高强度的覆盖，需要使用可变强度覆盖表。例如，令 $C=\{$OS, Web server, Database$\}$，则在表 2 的基础上增加表 3 中的 3 条测试用例，就可得到可变覆盖表 $VSCA(12;2,5,2^3 3^2,C)$。

表 3 覆盖表 $VSCA(12;2,5,2^3 3^2,C)$

No.	OS	Web server	Web brower	Connection type	Database
1	Windows	iPlanet	Chrome	PPP	Oracle
2	Windows	iPlanet	IE	ISDN	Access
3	Mac	Apache	IE	LAN	SQL

2.2 基本粒子群算法

粒子群优化算法模拟了鸟集群飞行觅食的行为，利用群体中的个体对信息的共享来达到最优目的，最早由 Eberhat 和 kennedy[9]于 1995 年提出。该算法将优化问题在搜索空间中的可能解表示为粒子在 n 维空间中的位置，通过粒子速度来控制粒子的飞行方向和距离，粒子 i 的速度和位置可分别用 $V_i=(v_{i,1},v_{i,2},\cdots,v_{i,n})$ 和 $X_i=(x_{i,1},x_{i,2},\cdots,x_{i,n})$ 表示。该算法采用迭代的方法，在解空间中通过追随自身的最佳位置(pBest)和整个种群的最佳位置(gBest)来不断搜索，直到找到最优解或有效解。通常采用适应度函数 $fitness(p)$ 来度量粒子的优劣，其返回值为测试用例 p 所能覆盖的组合数目。在第 $t+1$ 次迭代中，每个粒子的速度和位置更新公式如下：

$$v_{i,j}(t+1)=wv_{i,j}(t)+c_2r_2[gBest_{i,j}(t)-x_{i,j}(t)]+c_1r_1[pBest_{i,j}(t)-x_{i,j}(t)] \quad (1)$$

$$x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t+1) \quad (2)$$

其中，$v_{i,j}$ 表示第 i 个粒子在第 j 维的速度，$x_{i,j}$ 表示第 i 个粒子在第 j 维的位置；w 为惯性权值，表示粒子的历史速度对当前速度的影响；c_1 和 c_2 为加速因子，分别表示粒子对自身历史最优和整体历史最优的学习能力；r_1 和 r_2 是在[0,1]范围内服从均匀分布的两个随机数，用于保证种群的多样性。

在使用粒子群生成覆盖表的过程中，首先需要对搜索空间进行建模，粒子 i 的第 j 维位置 $x_{i,j}$ 对应于第 j 个因素，因此该位置的搜索空间为该因素 j 所对应的有效取值集 $D_j=\{1,2,\cdots,l_j\}$，即 $x_{i,j}\in D_j$。为了防止粒子在更新过程中飞出有效的搜索空间，需要对粒子位置进行边界处理，本文采取文献[10]建议的反射墙策略，即当粒子超越某一维边界时，使用式(3)对粒子进行反弹。

$$h(x_{i,j})=\begin{cases}2l_i-x_{i,j}+1, & \text{if } x_{i,j}>l_i\\ -x_{i,j}+1, & \text{if } x_{i,j}<1\\ x_{i,j}, & \text{else}\end{cases} \quad (3)$$

其次，为改善搜索结果，需要设定一个最大速度来限制粒子的速度，采用每个维度所对应因素取值范围的一半作为最大速度限制，即 $v_{max}=l_j/2$，粒子的速度区间为$[-v_{max},v_{max}]$。由于覆盖表生成问题属于离散组合问题，因此需要对原始 PSO 算法进行一定的改进：采用文献[10]提出的离散化方式，将粒子速度进行取整运算。结合以上策略，算法 1 给出了使用粒子群算法生成单个测试用例算法的伪代码。

算法 1　基于粒子群算法的单个测试用例生成算法

输入：群体规模 m，因素个数 n，各因素取值 D_i，待覆盖组合 S

输出：测试用例 gBest

```
1. NC=0;gBest=NULL;
2.   for(i=1;i<=m;i++)
3.   随机初始化每个粒子的位置矢量 Xi 和速度矢量 Vi;
4. While (NC<NCmax){
5. for(i=1;i<=m;i++){
6. 计算适应值 fitness(Xi);//返回 Xi 在 S 中覆盖的组合数目
7. if(fitness(Xi)==C(n,t)) return Xi;
8. if (fitness(Xi)>f(pBesti)) pBesti=Xi;
9. if (fitness(Xi)>f(gBest)) gBest=Xi;
10.  }
11. for(i=1;i<=m;i++){
12.   for(j=1;j<=n;j++){
13. 使用式(1)更新速度 Vi,j;
14.   if(Vi,j>lj/2) Vi,j=lj/2;
15. if(Vi,j<-lj/2) Vi,j=-lj/2;
16. 使用式(2)更新位置 Xi,j;
17. Xi,j=round(Xi,j);//对 Xi,j进行取整运算
18.   Xi,j=h(Xi,j);//使用式(3)对 Xi,j进行边界处理
19.       }
20.  }
21. NC++;//迭代次数
22. }
23. return gBest;
```

3　基于自适应粒子群的组合测试用例生成方法

3.1　方法的总体框架

组合测试方法通常根据因素间的交互作用来生成测试用例集，实现以较少的测试用例覆盖某些给定的因素取值组合，减少了不必要的测试开销成本。因此，在采用粒子群算法生成单条测试用例前，需要了解 SUT 中各因素的交互关系。本文提出的基于自适应粒子群的组合测试用例生成方法的总体框架如图 1 所示，分为测试环境构造模块和测试运行模块两个部分。

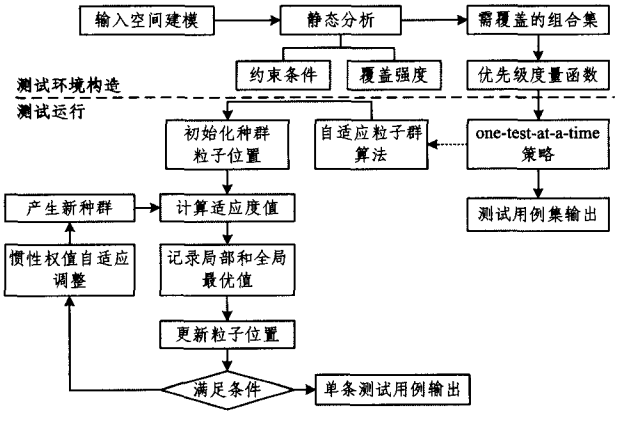

图 1　基于自适应粒子群的组合测试算法框架

(1)测试环境构造模块：通过静态分析的方法建模 SUT 的输入空间，综合考虑覆盖强度、约束条件等来获取需覆盖的所有因素的取值组合，每个组合均可对应一条缺省的测试用例，对每个组合进行优先级度量，优先选取权值大的组合用于单个测试用例的生成。

(2)测试运行模块：在测试环境构造模块的基础上，将选取的一个组合构造成仅有部分取值确定的测试用例，采用自适应粒子群算法完成单个测试用例的生成，即给缺省的参数选取合适的取值。在单条测试用例的生成过程中，采用不同的惯性权重来替代单一的惯性权重，根据粒子的当前状态来对每个粒子的惯性权重进行自适应调整，使粒子群算法更加适用于单条测试用例的生成。

3.2　one-test-at-a-time 策略

在组合测试用例生成问题中，one-test-at-a-time 策略由于具有简单、有效、便于扩展等特点而成为应用最普遍的方法之一。固定力度组合测试和可变力度组合测试均被证实是 NP-C 问题[11]，在使用 one-test-at-a-time 策略时，只能考虑使用多项式时间生成近似最优的测试用例，因此可将该策略与粒子群算法相结合用于覆盖表生成。其基本流程可概括为：首先生成一个空的测试用例集 TS，在需覆盖的组合集中选取一个组合，根据这个组合采用粒子群算法固定其他因素的取值，从而得到单个测试用例 t，移除测试用例 t 覆盖的组合并添加该测试用例 t 到 TS 中。当所有组合均被覆盖后，算法终止循环并返回 TS。

该策略在组合集中采用随机选取的方式选取一个组合，并未考虑每个组合的不同优先级，若能选取更加有效的组合，则可减少算法的迭代次数，使得算法的时间性能有所提升[12]。因此，有必要构造一个优先级度量函数，用于度量每个组合的优先级。

给定一个需覆盖的组合集，若某个组合生成的单条测试用例可覆盖更多的组合，则可一次性移除更多组合，从而能缩减下一步的计算量，显然该组合应被优先选择。例如，考虑一个简单的待测系统 $F=\{2^4\}$，当 one-test-at-a-time 策略运行至某一步时，其需覆盖组合集 *UncovCombset* 如表 4 所列，对于每个组合均可构造出一条对应的测试用例，其中“—”表示相应因素取值未确定。

表 4　需覆盖组合集及其对应的测试用例

No.	*UncovCombset*	对应的测试用例	包含的组合
1	$(f_2=1,f_3=2)$	(—,1,2,—)	1,2,5
2	$(f_1=1,f_2=1)$	(1,1,—,—)	1,2,5
3	$(f_2=2,f_3=2)$	(—,2,2,—)	3,5
4	$(f_1=1,f_2=2,f_3=1)$	(1,2,1,—)	4
5	$(f_3=2,f_4=1)$	(—,—,2,1)	1,2,3,5

通过表 4 中的数据可知，对于组合 $(f_1=1,f_2=2,f_3=1)$，其对应的测试用例不包含除自身外的其他 4 个组合，若优先选择该组合，就无法再覆盖 *UncovCombset* 中的任何组合，从而增加了下一步计算的难度。而对于组合 $(f_3=2,f_4=1)$，其所对应的测试用例可包含最多的组合，有利于生成一条组合覆盖率高的测试用例。因此，构造一个优先级度量函数 r_k，用于衡量集合 *UncovCombset* 中每个组合的优先级。

$$r_k=\sum_{i=1}^{s}P(c_i,t_k) \tag{4}$$

$$P(c_i,t_k)=\begin{cases}1, & \text{若 } t_k \text{ 包含 } c_i\\ 0, & \text{若 } t_k \text{ 不包含 } c_i\end{cases} \tag{5}$$

其中，r_k 表示第 k 个组合的权值，c_i 表示集合 *UncovCombset*

中第 i 个组合，t_k 表示组合 c_k 所对应的一条缺省的测试用例，s 表示集合 *UncovCombset* 所包含的组合数。

算法 2 one-test-at-a-time 策略

输入：需覆盖组合集 UncovCombset

输出：测试用例 TS

```
1. TS=NULL;r=0;S=NULL;
2. while(UncovCombet≠NULL){
3. n=count(UncovCombset);
4. for(i=1;i<=m;i++){
5. 根据式(4)、式(5)计算优先级 r_k;
6. if(r_k>r) {
7. r=r_k;S=UncovCombset_k;
8.     }
9.     }
10.    将 S 作为输入值，利用算法 1 生成测试用例 t;
11.    计算测试用例 t 覆盖的组合 Combet_t;
12.    UncovCombet=UncovCombet-Combet_t;
13. TS=TS∪{t};
14.  }
15. return TS;
```

3.3 自适应粒子群算法

对于启发式搜索算法，参数的选择对算法性能有巨大影响。已有研究表明，依据种群状态进行参数自适应调整能使种群演化更符合粒子群算法的假设，使算法的收敛性和解的正确率提高[13]。在粒子群算法本身具有良好鲁棒性的前提下使用自适应策略对参数进行调整，解决了参数选取困难的问题，能使粒子群算法更加适用于覆盖表的生成。

惯性权重表示粒子前一刻的速度对当前速度的影响，用于平衡粒子的局部搜索能力和全局搜索能力。惯性权重的设置对 PSO 算法的性能具有至关重要的作用。目前已有很多研究者针对惯性权重提出了很多改进策略[14]，包括线性递减、模糊规则和随机变化等。本文采用了一种在同一种群中使用不同惯性权值的调整策略，根据粒子的优劣对惯性权重进行自适应调整，以粒子与当前全局最优解(gBest)之间的距离作为粒子优劣的评价标准。由于组合测试问题属于离散型组合优化问题，相较于欧氏距离，采用曼哈顿距离进行距离度量更合适。已知两个向量 $X=\{x_1,x_2,\cdots,x_m\}$ 和 $Y=\{y_1,y_2,\cdots,y_m\}$，两者的曼哈顿距离可表示为：

$$d(X,Y)=\sum_{k=1}^{m}|x_k-y_k| \tag{6}$$

运用式(6)可计算出每个粒子与最优粒子之间的距离。为更好地度量粒子的优劣程度，进一步定义一个 f 值用于衡量粒子 X_i 与最优粒子之间的差异程度：

$$f(X_i)=\frac{d(X_i,X_{gBest})}{MaxDist} \tag{7}$$

其中，*MaxDist* 表示当前种群中与当前最优粒子 *gBest* 的最大距离。$f(X_i)$值越大，表示粒子 X_i 与最优粒子之间的差异越大，应该增大惯性权重，提高粒子的全局搜索能力以便更快找到最优解；反之，则需缩小惯性权重，提高粒子的局部搜索能力使其能够对周围进行细致的搜索。因此，建立符合上述规则的惯性权重调整模型：

$$w=w_{max}-(w_{max}-w_{min})e^{(f(X_i)/(f(X_i)-1))} \tag{8}$$

其中，w_{max}，w_{min}分别表示初始化时惯性权重 w 的最大值和最小值，本文根据文献[15]设置为：$w_{max}=0.9$，$w_{min}=0.4$，得到惯性权重 w 随 f 值的变化趋势，如图 2 所示。

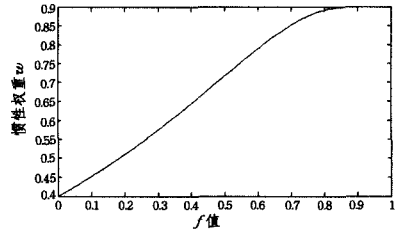

图 2 惯性权重 w 随 f 值变化的趋势

由图 2 可知，w 的取值范围为[0.4,0.9]，其取值随着 f 值单调递增，显然，该惯性权重调整模型符合要求。

4 实验分析

为了验证本文提出的改进策略的有效性，将优先级度量函数分别与原始粒子群算法和自适应粒子群算法相结合，采用 MATLAB 编程实现与之对应的 P-PSO 算法和 AP-PSO 算法，将这两种算法与 PSO 算法进行实验对比。本文采用 15 个具有代表性的实例用于实验分析，涉及到覆盖矩阵、混合覆盖矩阵和可变力度覆盖矩阵各 5 组实例。

表 5 实验采用的 15 个覆盖表

CA	MCA	VSCA
CA1(2,4,3)	MCA6(2,9,$8^2$$7^3$$5^2$$3^2$)	VSCA11(2,15,3,CA(3,3^4))
CA2(2,10,10)	MCA7(2,30,$4^6$$3^{14}$$2^{10}$)	VSCA12(2,10,$4^3$$3^3$$2^4$, CA(3,$4^3$))
CA3(2,20,10)	MCA8(3,10,$5^1$$3^4$$2^5$)	VSCA13(2,15,$5^3$$4^5$$3^7$, MCA(3,$5^3$))
CA4(3,10,4)	MCA9(4,5,$5^2$$4^3$)	VSCA14(3,10,3, MCA(4,3^4))
CA5(4,8,5)	MCA10(4,8,$8^2$$7^2$$6^2$$5^2$)	VSCA15(3,15,$4^5$$2^{10}$, MCA(3,$2^5$))

由于粒子群算法中存在随机因素的影响，因此在实证研究中针对每组实验数据均独立运行 20 次，并取其平均值作为对比数据。本文算法的参数设置参照文献[16]：最大迭代次数 $NC_{max}=1000$，学习因子 $c_1=c_2=2$。

表 6 从测试用例集规模和算法运行时间两个方面列出了 3 种算法针对 15 个实例生成测试用例集的效果。

表 6 PSO，P-PSO 和 AP-PSO 算法的比较

Covering Array	PSO		P-PSO		AP-PSO	
	size	time/s	size	time/s	size	time/s
CA1	9	10.9	9	12.2	9	11.8
CA2	159.9	232	152.7	186	151.4	91
CA3	221.5	1123	195.7	930	194.4	657
CA4	42.3	322	40.8	283	40.9	192
CA5	1373.1	4234	1250.2	3866	1241.6	3432
MCA6	117.4	144	109.1	106	111.5	58
MCA7	120.3	175	119.3	168	119.2	126
MCA8	47.2	124	42.6	105	44.7	79
MCA9	93.5	46	70.4	43	69.8	25
MCA10	4255.4	19613	3597.6	17821	3637.1	15312
VSCA11	33.6	29	32.5	30	32.7	32
VSCA12	65.6	158	62.2	129	61.3	54
VSCA13	241.3	903	216.2	714	214.5	336
VSCA14	68.6	135	53.5	108	54.9	114
VSCA15	125.9	612	120.1	579	121.5	350

从测试用例集规模上看，除较为简单的覆盖表 CA1 外，P-PSO 算法和 AP-PSO 算法均优于 PSO 算法，尤其针对较为复杂的覆盖表具有明显的优势，例如 CA5，MCA9 和 MCA10 等。对比 P-PSO 算法和 AP-PSO 算法，两者在测试用例集规模上无明显差距，由此得出，本文提出的优先级度量函数对缩减测试用例集规模有一定的效果。

从时间性能上看，P-PSO 算法略优于 PSO 算法，AP-PSO 算法相较于 PSO 算法和 P-PSO 算法，除 CA1，VSCA11 和 VSCA14 外，均具有较为显著的优势，其节约的时间达到 16%～63%。由此可见，本文提出的自适应策略可有效减少算法执行时间。

综上所述，相较于原始粒子群算法，本文提出的改进算法在生成测试用例集规模和算法执行时间上具有一定的优势。

结束语　目前，组合测试领域较多地关注于相对简单的固定力度组合测试用例模型。本文提出了一种基于自适应粒子群的组合测试用例生成算法，该算法具有广泛的适用性，可用于生成任意覆盖强度的覆盖表。针对粒子群算法易受配置参数影响的问题，根据粒子优劣对惯性权值进行自适应调整，进一步提升了算法性能。实验验证了所提算法相较于原始粒子群算法的优势。

由于粒子群算法具有易陷入局部最优、后期收敛速度慢等缺陷，已有很多研究者开展了一系列的改进研究工作，提出了各种变种粒子群算法。在未来的工作中，可考虑对这些算法进行研究，以寻找到更适合覆盖表生成的改进算法。

参 考 文 献

[1] KUHN D R, WALLACE D R, GALLO A M. Software fault Interactions and implications for software testing[C]// IEEE Trans. on Software Engineering. 2004: 418-421.

[2] XIE X Y, XU L, XU B W, et al. Survey of Evolutionary Testing [J]. Frontiers of Computer Science & Technology, 2008, 2(5): 449-466. (in Chinese)
谢晓园，许蕾，徐宝文，等. 演化测试技术的研究[J]. 计算机科学与探索，2008，2(5)：449-466.

[3] SUN W W, JIANG J, NIE C H. Configurable Hybrid Algorithm for Combinatorial Test Suite Generation[J]. Computer Science, 2011, 38(8): 130-135. (in Chinese)
孙文雯，蒋静，聂长海. 一种组合测试用例生成的可配置混合算法[J]. 计算机科学，2011，38(8)：130-135.

[4] CHEN X, GU Q, WANG Z Y, et al. Framework of Particle Swarm Optimization Based Pairwise Testing[J]. Journal of Software, 2011, 22(12): 2879-2893. (in Chinese)
陈翔，顾庆，王子元，等. 一种基于粒子群优化的成对组合测试算法框架[J]. 软件学报，2011，22(12)：2879-2893.

[5] MAO C Y, YU X X, XUE Y Z. 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)
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.

[6] ZHA R J, ZHANG D P, NIE C H, et al. Test Data Generation Algorithms of Combinatorial Testing and Comparison Based on Cross-Entropy and Particle Swarm Optimization Method[J]. Journal of Computers, 2010, 33(10): 1896-1908. (in Chinese)
查日军，张德平，聂长海，等. 组合测试数据生成的交叉熵与粒子群算法及比较[J]. 计算机学报，2010，33(10)：1896-1908.

[7] WU H Y, NIE C H, LIANG Y L, et al. A Discrete Particle Swarm Optimization for Covering Array Generation[J]. IEEE Transactions on Evolutionary Computation, 2015, 19(4): 575-591.

[8] KUHN D, REILLY M. An investigation of the applicability of design of experiments to software testing[C]// Proc. of the 27th Annual NASA Goddard/IEEE Software Engineering Workshop. NASA Goddard Space Flight Center, 2002: 1-5.

[9] KENNEDY J, EBERHART R. Particle swarm optimization [C]// Proc of the 4th IEEE Int Conf on Neural Networks. Piscataway, NJ: IEEE, 1995: 1942-1948.

[10] PARSOPOULOS K E, VRAHATIS M N. Recent approaches to global optimization problems through particle swarm optimization[J]. Natural Computing, 2002, 116(2/3): 235-3063.

[11] WANG Z Y, QIAN J, CHEN L, et al. Generating Variable Strength Combinatorial Test Suite with One-test-at-a-time Strategy[J]. Chinese Journal of Computer, 2012, 35(12): 2541-2552. (in Chinese)
王子元，钱巨，陈林，等. 基于 One-test-at-a-time 策略的可变力度组合测试用例生成方法[J]. 计算机学报，2012，35(12)：2541-2552.

[12] ZHANG N, YAO L, BAO X A, et al. Multi-Objective Optimization Based On-Line Adjustment Strategy of Test Case Prioritization[J]. Journal of Software, 2015(10): 2451-2464. (in Chinese)
张娜，姚澜，包晓安，等. 多目标优化的测试用例优先级在线调整策略[J]. 软件学报，2015(10)：2451-2464.

[13] ZHEN Z H, ZHANG J, LI Y, et al. Adaptive particle swarm optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2009, 39(6): 1362-1381.

[14] LI S F, LI T Y. Distance-based Adaptive Fuzzy Particle Swarm Optimization[J]. Computer Science, 2011, 38(8): 257-259. (in Chinese)
李朔枫，李太勇. 一种基于距离的自适应模糊粒子群优化算法[J]. 计算机科学，2011，38(8)：257-259.

[15] YOU B, CHEN G, GUO W. A Discrete PSO-Based Fault-Tolerant Topology Control Scheme in Wireless Sensor Networks[C]// Proceedings of the 5th International Conference on Advances in Computation and Intelligence. 2010: 1-12.

[16] SU J S, GUO W Z, YU C L, et al. Fault-Tolerance Clustering Algorithm with Load-Balance Aware in Wireless Sensor Network[J]. Chinese Journal of Computer, 2014, 37(2): 445-456. (in Chinese)
苏金树，郭文忠，余朝龙，等. 负载均衡感知的无线传感器网络容错分簇算法[J]. 计算机学报，2014，37(2)：445-456.