组合测试数据生成的交叉熵与粒子群算法及比较

查日军[1),2)] 张德平[4)] 聂长海[2),3)] 徐宝文[2),3)]

[1)](东南大学计算机科学与工程学院 南京 210096)

[2)](南京大学软件新技术国家重点实验室 南京 210093)

[3)](南京大学计算机科学与技术系 南京 210093)

[4)](南京航空航天大学 信息科学与技术学院 南京 210016)

摘 要 测试数据生成是组合测试的一个关键问题.文中提出以数理统计为基础的交叉熵方法和以仿生学为基础的粒子群优化算法来生成两两组合测试数据,交叉熵方法采用最优选择概率产生测试数据,而粒子群算法则在可行解空间中搜索具有最优适应值的测试数据.文章给出了交叉熵方法最优选择概率的理论推导,并对两种算法所生成的测试数据集进行约简.将两种算法和现有的贪心方法、代数方法及其它启发式搜索方法进行比较,实验表明交叉熵方法和粒子群算法具有一定的优势和特点.

关键词 软件测试;组合测试;交叉熵;粒子群优化

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2010.01896

Test Data Generation Algorithms of Combinatorial Testing and Comparison Based on Cross-Entropy and Particle Swarm Optimization Method

ZHA Ri-Jun[1),2)] ZHANG De-Ping[4)] NIE Chang-Hai[2),3)] XU Bao-Wen[2),3)]

[1)](*School of Computer Science & Engineering, Southeast University, Nanjing* 210096)

[2)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093)

[3)](*Department of Computer Science and Technology, Nanjing University, Nanjing* 210093)

[4)](*College of Information Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing* 210016)

Abstract The test suite generation is one of key issues of combinatorial testing. This paper uses Cross-Entropy method of statistics and Particle Swarm Optimization from bionics to generate pairwise test data of combinatorial testing. The cross-entropy method used the best selection probability to generate test data and the particle swarm optimization generates test data by searching one with best fitness in a feasible solution. A theoretical method is given for the optimal probability selection of the cross-entropy method and a reduction technique is proposed to reduce the test suite generated by two methods. The empirical results show that the cross-entropy method and the particle swarm optimization have some merits compared with existing greedy algorithms, algebra methods and other heuristic search algorithms.

Keywords software testing; combinatorial testing; cross-entropy; particle swarm optimization

---

收稿日期:2010-08-22.本课题得到国家自然科学基金(90818027,60721002,60873050,60773104)、国家"八六三"高技术研究发展计划项目基金(2008AA01Z143,2009AA01Z147)、国家"九七三"重点基础研究发展规划项目基金(2009CB320703)及上海市科委重点实验室基金(09DZ2272600)资助.**查日军**,男,1971 年生,博士研究生,讲师,主要从事软件测试技术、软件可靠性等方面的科研工作. E-mail: zharijun@seu.edu.cn. **张德平**,男,1973 年生,博士,讲师,主要从事软件测试技术、软件可靠性、数理统计等方面的教学、科研工作. **聂长海**,男,1971 年生,博士,副教授,主要从事软件测试技术、模糊信息处理等方面的教学与科研工作. **徐宝文**,男,1961 年生,博士,教授,博士生导师,主要从事程序设计语言、软件工程、并行与网络软件等方面的教学与科研工作.

1 引　言

软件测试是软件开发过程中的一个重要环节，是保证软件产品质量的重要手段.由于穷尽测试在实际测试过程中有时不可能或没必要，因此如何生成数量少质量高的测试数据集是人们关注的焦点.组合测试是一种科学有效的软件测试方法，该方法根据实际需要，用尽量少的测试数据尽可能多地覆盖软件系统中的各因素及相应组合，系统检测它们之间的相互作用对系统所产生的影响.根据覆盖程度的不同组合覆盖可分为单因素覆盖、两两组合覆盖、三三组合覆盖等高维多因素组合覆盖.两两组合覆盖已在软件测试领域中得到了广泛的应用，人们在应用两两组合覆盖方法对软件系统进行测试时发现了很多传统方法难以发现的错误[1]，研究发现大约有 70%的软件故障是由一个或两个参数的相互作用而引发[2]，因而两两组合覆盖测试技术的研究具有重要意义，本文讨论的组合覆盖测试是两两组合覆盖测试.

由于最小两两组合覆盖的测试数据生成问题是一个 NP-hard 问题[3]，因而人们尝试用各种启发式算法与代数构造方法，生成规模尽可能小的测试数据集[1-15].早年，人们多使用正交实验设计方法来生成测试数据[4]，该方法以正交表为基础，而正交表在其存在性和构造方法等方面还存在很多未解决的难题.此后，Kobayashi 等[5]和 Williams[6]分别给出了两种代数方法，可以有效地生成测试数据.除了代数方法外，美国贝尔实验室的 Cohen 等[7]提出了一种测试数据的启发式生成方法，开发了相应的测试数据自动生成系统 AETG，并申请了专利.美国喷气推进实验室的 Tung 等[8]提出了另外一种基于贪心策略的两两组合测试数据集启发式生成算法，并开发了一种名为 TCG 的测试数据生成工具. Lei 等[3]提出了一种基于参数顺序的渐进扩充的两两组合测试覆盖测试数据生成方法，也开发了相应的测试数据自动生成系统 PAIRTEST.

近年来，国内在组合测试数据生成领域也进行了系统的研究，提出了多种数学与启发式生成算法.中国科学院的严俊等基于 SAT 求解工具以及回溯法生成两两组合覆盖测试数据[9]；南京大学的陈翔等基于蚁群算法生成变力度的交互组合测试数据[10]；这几年我们在这一领域内同样作了相关深入的研究，首先提出了一种基于网络图的两两组合测试数据生成算法[11]，并对 AETG 方法进行拓展，给出了一种支持高维组合测试数据集生成方法[12].在组合覆盖测试模型的基础上，将所有可用测试数据表示为一棵解空间树，利用回溯法搜索解空间树，选择出一个规模最小的路径集，以实现对测试参数的两两组合覆盖[13].利用组合分析的方法，针对二水平多因素系统给出两两组合测试数据生成算法[14]，对一类只在相邻因素之间存在相互作用的系统，提出了相邻因素组合测试的概念，给出了相邻因素两两组合覆盖表的生成算法[15].

作为数学方法和启发式搜索方法的重要补充，本文分别利用交叉熵方法和粒子群算法提出了两种两两组合测试数据生成方法.基于两两组合覆盖测试模型，首先，根据交叉熵方法的应用特点，从理论上推导出组合测试数据的最优选择概率，并成功应用于组合测试数据的生成；其次将以仿生学为基础的粒子群算法应用于组合测试数据生成，针对其搜索容易落入局部最优的不足，应用一种扰动机制扩大其搜索范围，有效提高组合测试数据生成质量.进一步，基于启发式搜索方法生成组合测试数据的特点，对这两种算法所生成的测试数据集进行约简；最后通过实验对比与灵敏度分析，与已有的启发式搜索方法进行比较，结果表明利用交叉熵方法和粒子群算法生成的组合测试数据在规模、时间上具有其优势和特点.

本文第 2 节介绍组合覆盖测试的基本概念与模型；第 3 节提出以数理统计为基础的交叉熵方法在两两组合测试数据生成上的应用；第 4 节给出以仿生学为基础的粒子群算法来生成两两组合测试数据集；第 5 节给出由两种算法生成测试数据集的约简算法；第 6 节分析实验数据将交叉熵方法、粒子群算法及现有的相关方法进行比较；最后总结并提出进一步的研究工作.

2 组合覆盖测试模型

设影响待测软件 SUT(Software Under Test)的参数(因素)有 n 个，形成有限集 $P=(f_1,f_2,\cdots,f_n)$，这些参数可以是 SUT 的配置参数、内部事件、外部输入等.不妨设 SUT 的每个参数 f_i 可在有限离散点集 V_i 中取值，V_i 中有 $k(|V_i|=k,1\leq i\leq n)$ 个元素，记为 $V_i=\{V_{i1},V_{i2},\cdots,V_{ik}\}$，$k$ 称为参数 V_i 的水平数.进一步假定这些参数是相互独立的，即某个参数的具体取值不会影响到其它参数的取值或

存在性.

定义 1. 如果 $x_1 \in V_1, x_2 \in V_2, \cdots, x_n \in V_n$,则称 n 元组$(x_1, \cdots, x_n)$为待测软件 SUT 的一条测试数据.

定义 2. 设 $\boldsymbol{A}=(x_{ij})_{m\times n}$为一个 $m \times n$ 的矩阵,其第 j 列表示 SUT 的参数 f_j,第 j 列的所有元素均取自集合 $V_j(j=1,2,\cdots,n)$,即 $x_{ij} \in V_j$. 如果 $\boldsymbol{A}$ 中任意两列第 i 列和第 j 列都满足:f_i 和 f_j 符号的全部两两组合都在第 i 列和第 j 列形成的二元有序对中等概率出现,则称 $\boldsymbol{A}$ 为正交表. 如果 $\boldsymbol{A}$ 的任意两列第 i 列和第 j 列都满足:f_i 和 f_j 符号的全部两两组合都在第 i 列和第 j 列形成的二元有序对中出现(不要求等概率出现),则称 $\boldsymbol{A}$ 是一个两两组合覆盖表. $\boldsymbol{A}$ 中的每一行均表示一条测试数据,m 为测试数据集中测试数据的数量. 对于某一个两两组合覆盖表 $\boldsymbol{A}$,如果 m 是能够保证上述条件成立的最小正整数,则称 $\boldsymbol{A}$ 为最小两两组合覆盖表.

两两组合覆盖的测试数据生成问题就是两两组合覆盖表的构造问题,由于最小两两组合覆盖表的生成问题是一个 NP-hard 问题[3],在实际的测试中,人们一般都利用启发式算法、贪心算法和一些数学代数等方法近似求解. 这些方法具有不同的优缺点,针对某些具体问题有的可以产生最优解. 作为已有方法的重要补充,本文提出了以数理统计为基础的交叉熵方法和以仿生学为基础的粒子群算法生成两两组合测试数据集的启发式方法,该方法利用交叉熵和粒子群算法每次生成一条最优测试数据放入测试数据集中,直到将组合覆盖集中所有参数的两两组合对都完全覆盖为止. 算法的基本框架由算法 1 给出.

算法 1. 两两覆盖组合测试数据集的生成.

初始化测试数据集 Ts 为空集;

根据系统初始化组合覆盖集 $CombSet$;

While ($CombSet \neq \varnothing$)

  由交叉熵和粒子群算法选择生成一个最优测试数据 t_i, 加入到测试数据集 Ts 中;

  更新 $CombSet$,删除由测试数据 t_i 覆盖的所有两两组合对;

End While

3 基于交叉熵方法的组合测试数据生成

交叉熵(Cross Entropy, CE)方法最早由 Rubinstein[16]引入用来估计复杂随机网络中稀有事件发生的概率. 该方法通过交叉熵的简单变形解决稀有事件发生概率的估计问题,是一种使估计的方差达到最小的适应性算法. 其主要思想是将"确定性"优化问题转变成一个伴随"随机"优化问题,然后用稀有事件的随机模拟技术来求解此"随机"优化问题,人们已经成功地应用此方法解决了一些很难求解的组合优化问题[17].

下面我们讨论将交叉熵方法运用于两两覆盖的组合测试数据生成的问题. 应用交叉熵方法关键需要解决两个问题:如何产生随机样本和在每一次迭代中如何修正概率参数.

当我们把组合测试数据当作是样本,即是组合测试数据的随机生成问题. 这里我们可将组合测试数据的生成转化为一个有向图的遍历来解决,如图 1所示.

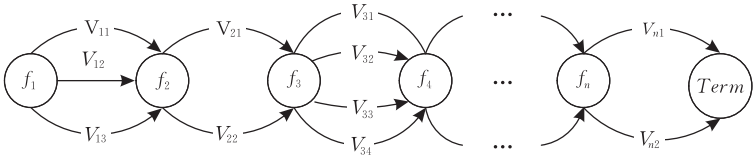

图 1 搜索空间

图中各节点表示各个不同的参数 $f_1, f_2, \cdots, f_n$, $f_i(1 \leq i \leq n)$,每一条出边分别表示参数 f_i 的一个可能取值 V_{ij}. 因此,从节点 f_1 到虚拟终端节点 $Term$ 之间存在一条路径,则该路径就是一条有效的组合测试数据. 因此组合测试数据的随机生成,可以看成从 f_1 到 f_n 的每一个节点 f_i 根据某个概率分布选择一条出边,从而形成一条组合测试数据. 概率参数的修正则需通过使测试数据两两组合覆盖率最大转化为与之相关的伴随随机优化问题来解决.

令 $\boldsymbol{X}=(X_1, X_2, \cdots, X_n)$为某个概率空间 χ 的一个随机向量,表示组合测试数据生成过程中根据某个概率分布生成的一条组合测试数据,其中随机变量 $X_i(i=1,2,\cdots,n)$表示按选择概率 $P_i=(p_{i1}, p_{i2}, \cdots, p_{ik})$在第 i 个参数 f_i 的 $k(k=|V_i|)$个可能取值中选取的一个取值,即 $X_i(i=1,2,\cdots,n)$以概率 p_{i1} 选取 V_{i1},以概率 p_{i2} 选取 V_{i2}, 以概率 p_{ik} 选取

V_{ik}，并且有$\sum_{l=1}^{k} p_{il}=1$，则由 $\boldsymbol{X}$ 可确定一条测试数据 $\boldsymbol{x}$. 一条最优测试数据 $\boldsymbol{x}$ 可定义为其在覆盖表中覆盖其它数据未覆盖的参数取值对数最多，或其在覆盖表中覆盖其它数据已覆盖的参数取值对数最少. 若记测试数据 $\boldsymbol{x}$ 在覆盖表中覆盖其它数据已覆盖的参数取值对数为 $S(\boldsymbol{x})$. 则一条最优测试数据 $\boldsymbol{x}$ 的生成应为其 $S(\boldsymbol{x})$ 在概率空间 χ 上取最小值，即

$$S(\boldsymbol{x}^*)=\gamma^*=\min_{x\in\chi} S(\boldsymbol{x}) \tag{1}$$

事件在概率空间 χ 上发生. 令 $\boldsymbol{x}$ 服从的选择概率分布为 $f(\cdot;\boldsymbol{u})$，则上述式(1)可转化为如何确定选择概率分布 $f(\cdot;\boldsymbol{u})$，使得事件 $S(\boldsymbol{x}^*)=\gamma^*=\min_{x\in\chi} S(\boldsymbol{x})$ 发生为最优. 定义$\{\boldsymbol{I}_{\{S(\boldsymbol{x})\le\gamma\}}\}$为 χ 上不同 $\gamma(\in R)$ 值的示性函数集合，则式(1)中的最优选择概率 $f(\cdot;\boldsymbol{u})$ 的确定问题可转化为与之相伴随的概率估计问题来求解[16]：

$$l(\gamma)=\boldsymbol{P}_{\boldsymbol{u}}(S(\boldsymbol{X})\le\gamma)=\sum_{\boldsymbol{x}}\boldsymbol{I}_{\{S(\boldsymbol{x})\le\gamma\}} f(\boldsymbol{x};\boldsymbol{u})=\boldsymbol{E}_{\boldsymbol{u}}\boldsymbol{I}_{\{S(\boldsymbol{x})\le\gamma\}} \tag{2}$$

其中 $\boldsymbol{P}_{\boldsymbol{u}}$ 和 $\boldsymbol{E}_{\boldsymbol{u}}$ 分别为相对于最优选择概率 $f(\cdot;\boldsymbol{u})$ 的概率测度和期望.

事实上，当 $\gamma=\gamma^*$ 时，用于估计 l 的最优抽样分布就是需要确定的最优选择概率. 如，假定当 $\gamma=\gamma^*$ 并且 $f(\cdot;\boldsymbol{u})$ 为 $\boldsymbol{X}$ 上均匀分布时，由式(2)得 $l(\gamma^*)=f(\boldsymbol{x}^*;\boldsymbol{u})=1/|\boldsymbol{X}|$，而 $f(\boldsymbol{x}^*;\boldsymbol{u})$ 正是估计 $l(\gamma^*)$ 的最佳抽样分布. 因此，最优选择概率的确定问题转化为确定估计 l 的最优抽样分布.

当 $\gamma=\gamma^*$ 时 $l(\gamma)$ 估计的最直接方法是采用重要抽样方法[18]：根据 $\boldsymbol{X}$ 上的概率分布 g 抽取样本 $\boldsymbol{x}_1$，$\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N$，则 l 的估计为

$$\hat{l}=\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{I}_{\{S(\boldsymbol{x}_i)\le\gamma\}}\frac{f(\boldsymbol{x}_i;\boldsymbol{u})}{g(\boldsymbol{x}_i)} \tag{3}$$

显然，当概率分布 g 取

$$g^*(\boldsymbol{x})=\frac{\boldsymbol{I}_{\{S(\boldsymbol{x})\le\gamma\}} f(\boldsymbol{x};\boldsymbol{u})}{l} \tag{4}$$

时，只需抽取一个样本，即 $N=1$ 就可得到 l 的一个方差为零的无偏估计形式

$$\boldsymbol{I}_{\{S(\boldsymbol{x}_i)\le\gamma\}}\frac{f(\boldsymbol{x}_i;\boldsymbol{u})}{g^*(\boldsymbol{x}_i)}=l \tag{5}$$

从式(5)可以看出，g^* 依赖于未知参数 l，很难确定. 因此一般采用在概率分布簇$\{f(\cdot;\boldsymbol{v})\}$中选取概率分布 g 来解决这个问题，即确定推断参数 $\boldsymbol{v}$ 使得 $f(\cdot;\boldsymbol{v})$ 与概率分布 g^* 差别达到最小. 常用来衡量两个概率分布相似程度的测度是 $K-L$ 距离和交叉熵[18].

这样，最优选择概率的确定问题就转化为确定推断参数 $\boldsymbol{v}$，使得 $f(\cdot;\boldsymbol{v})$ 与概率分布 g^* 的交叉熵最小，即

$$\boldsymbol{v}^*=\arg\min_{\boldsymbol{v}}-\int g^*(\boldsymbol{x})\ln f(\boldsymbol{x};\boldsymbol{v})\mathrm{d}x \tag{6}$$

将式(4)代入式(6)可得如下等价形式的推断参数确定形式：

$$\boldsymbol{v}^*=\arg\max_{\boldsymbol{v}}\boldsymbol{E}_{\boldsymbol{u}}\boldsymbol{I}_{\{S(\boldsymbol{X})\le\gamma\}}\ln f(\boldsymbol{X};\boldsymbol{v}) \tag{7}$$

则最优推断参数 $\boldsymbol{v}^*$ 估计可由下式给出：

$$\hat{\boldsymbol{v}}^*=\arg\max_{\boldsymbol{v}}\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{I}_{\{S(\boldsymbol{x}_i)\le\gamma\}}\ln f(\boldsymbol{x}_i;\boldsymbol{v}) \tag{8}$$

这里样本 $\boldsymbol{x}_i$ 是由概率分布 $f(\cdot;\boldsymbol{u})$ 抽样产生.

由于 $\boldsymbol{x}_i$ 是根据概率分布 $\boldsymbol{P}$ 生成，因此概率参数向量 $\boldsymbol{P}$ 就是推断参数 $\boldsymbol{v}$，其概率分布具有如下形式

$$f(\boldsymbol{x}_i;\boldsymbol{P})=\prod_{l=1}^{n}P\{X_{il}=x_{ilj}\},\ x_{ilj}\in V_l \tag{9}$$

其中 x_{ilj} 表示第 i 个组合测试数据样本 $\boldsymbol{x}_i$ 中第 l 个参数的取值，$P\{X_{il}=x_{ilj}\}$ 表示第 i 个测试用例集样本中第 l 个参数选择 V_{lj} 的概率，并且满足$\sum_{j=1}^{k}P\{X_{il}=x_{ilj}\}=1$，$k=|V_l|$. 则由拉格朗日乘子法，式(8)取最大值只需满足

$$\frac{\partial}{\partial P\{X_{il}=x_{ijl}\}}\Big[\sum_{i=1}^{N}\boldsymbol{I}_{\{S(\boldsymbol{x}_i)\le\gamma\}}\ln f(\boldsymbol{x}_i;\boldsymbol{P})+\sum_{l}u_l\cdot\Big(\sum_{j=1}^{k}P\{X_{il}=x_{ilj}\}-1\Big)\Big]=0 \tag{10}$$

其中 u_l 为拉格朗日因子，由此可得

$$P\{X_{il}=x_{ilj}\}=\frac{\sum_{i=1}^{N}\boldsymbol{I}_{\{S(\boldsymbol{x}_i)\le\gamma\}}\cdot\boldsymbol{I}_{\{x_i\in\chi_{lj}\}}}{\sum_{i=1}^{N}\boldsymbol{I}_{\{S(\boldsymbol{x}_i)\le\gamma\}}},\quad l=1,2,\cdots,n,\ j=1,2,\cdots,|V_l| \tag{11}$$

其中 $\boldsymbol{I}_{\{x_i\in\chi_{li}\}}$ 表示测试数据 $\boldsymbol{x}_i$ 中的第 l 个参数选择了值 V_{lj}，这样就得到了一个组合测试数据中不同参数取值选择概率的修正公式.

式(11)的直观含义为：第 l 参数各个不同取值的选择概率为所有目标函数值(覆盖组合覆盖表中其它测试数据已覆盖的数据对个数)不大于 γ 的组合测试数据中，选择 V_{lj} 的总次数与测试数据数的比值. 由此可知，在每个子域中的测试用例选择概率的修正都是选用最有利于接近最优目标的“精英”样本来修正，这样修正后使得下一次生成的测试用例集更接近于最优测试目标，这种机制会很容易找到最优选择概率.

在交叉熵方法的迭代优化过程中,需要解决的问题是如何确定 γ,使得 γ 的值尽可能地接近最优值 γ^*,并且当 γ 接近 γ^* 时,事件 $\{S(\boldsymbol{X}_i)\leq\gamma\}$ 发生的概率不能太小,否则在测试用例集生成过程中就很少观察到事件 $\{S(\boldsymbol{X}_i)\leq\gamma\}$ 发生.解决这个问题的主要想法是通过抽样产生一个推断参数序列 $\{\boldsymbol{v}_t,t\geq 0\}$ 和不同水平的 $\{\gamma_t,t\geq 1\}$ 序列.在每一次抽样中(N 个组合测试数据样本),参数 $\boldsymbol{v}_t$ 的修正由式(11)给出,γ_t 的选择由 N 个样本计算出的 N 个目标函数值的 ρ 分位点给出,这里 ρ 不是一个非常小的实数,如 $\rho=0.02$,这样可保证事件 $\{S(\boldsymbol{X}_i)\leq\gamma_t\}$ 发生的概率至少为 ρ.具体由一个两步迭代过程实现:

1. 适应修正 γ_t. 固定 $\boldsymbol{v}_{t-1}$,令 γ_t 为在参数 $\boldsymbol{v}_{t-1}$ 下的目标函数值序列 $S(\boldsymbol{X})$ 的 $\rho\cdot 100\%$ 分位数.即 γ_t 满足

$$P_{\boldsymbol{v}_{t-1}}(S(\boldsymbol{X})\leq\gamma_t)\leq\rho \text{ 并且 } P_{\boldsymbol{v}_{t-1}}(S(\boldsymbol{X})\geq\gamma_t)\leq 1-\rho,$$

其中 $\boldsymbol{X}\sim f(\cdot;\boldsymbol{v}_{t-1})$.最简单的估计是根据 $f(\cdot;\boldsymbol{v}_{t-1})$ 抽样出 N 个组合测试数据样本 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N$ 计算出目标函数样本值,并将其按从小到大排列:$S_{(1)}\leq S_{(2)}\leq\cdots\leq S_{(N)}$,最后计算目标函数样本值的 $\rho\cdot 100\%$ 来估计 γ_t,即 $\hat{\gamma}_t=S_{(\lceil\rho\cdot N\rceil)}$.

2. 适应修正 $\boldsymbol{v}_t$. 固定 γ_t 和 $\boldsymbol{v}_{t-1}$,由式(11)得到修正的 $\bar{\boldsymbol{v}}_t$,并采用平滑技术得到 $\hat{\boldsymbol{v}}_t$ 的修正值:

$$\hat{\boldsymbol{v}}_t=\alpha\bar{\boldsymbol{v}}_t+(1-\alpha)\hat{\boldsymbol{v}}_{t-1},\ \alpha\in(0.4,0.9).$$

这样,在首次循环中,初始化在每个节点(因素) $i(i=1,2,\cdots,n)$ 选择各个水平取值的概率为等概分布,即节点 i 的每个不同水平取值均以概率 $1/|V_i|$ $(i=1,2,\cdots,n)$ 选择,由上面的过程可得到 $\hat{\gamma}_t$ 和 $\hat{\boldsymbol{v}}_t$,如此循环,可得到序列对 $\{(\hat{\gamma}_t,\hat{\boldsymbol{v}}_t),t=1,2,\cdots\}$,由文献[19]知 $\hat{\gamma}_t\to\gamma^*$,于是一条最优测试数据的生成过程由如下算法给出.

算法 2. 最优选择概率生成.

1. 初始化每个节点各个水平的选择概率 $\hat{\boldsymbol{v}}_0$ 为等概分布,令 $t=1$;

2. 根据概率分布 $f(\cdot;\hat{\boldsymbol{v}}_{t-1})$ 生成 N 个组合测试数据样本 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N$,计算各个测试数据的目标函数样本值(见算法 3),并排序 $S_{(1)},\cdots,S_{(N)}$,找出样本 $\rho\cdot 100\%$ 分位数 $\hat{\gamma}_t=S_{(\lceil\rho\cdot N\rceil)}$;

3. 用同样的组合测试数据样本 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N$,由式(11)解出 $\bar{\boldsymbol{v}}_t$,应用平滑技术得到 $\hat{\boldsymbol{v}}_t$:

$$\hat{\boldsymbol{v}}_t=\alpha\bar{\boldsymbol{v}}_t+(1-\alpha)\hat{\boldsymbol{v}}_{t-1};$$

4. 令 $t=t+1$,重复步 2 和 3,直到满足停止条件(如:对于某个给定的 d 有 $\hat{\gamma}_t=\hat{\gamma}_{t-1}=\cdots=\hat{\gamma}_{t-d}$,或 $\hat{\boldsymbol{v}}_t$ 不再变化,或达到最大迭代步数).

输出:最优选择概率 $\hat{\boldsymbol{v}}_t$.

在算法 2 中由概率分布 $f(\cdot;\hat{\boldsymbol{v}}_{t-1})$ 生成组合测试数据样本 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N$ 的具体实现过程由算法 3 给出.

算法 3. 组合测试数据样本的生成.

输入:$f(\cdot;\hat{\boldsymbol{v}}_{t-1})$

For ($i=1$ to N):

1. Repeat until (当前节点为虚拟终端节点 *Term*)
根据概率分布 $f(\cdot;\hat{\boldsymbol{v}}_{t-1})$ 在每个节点(因素)选择一个水平值;

2. 计算目标函数样本值 $S(\boldsymbol{X}_i)$;

输出:每条组合测试数据的目标函数值 $S(\boldsymbol{X}_i)$.

我们根据算法 2 生成最优选择概率 $\hat{\boldsymbol{v}}_t$,再利用算法 3 生成 N 个测试数据,选择出一条最优的组合测试数据.其算法复杂度为 $O(T\cdot N\cdot C_n^2\cdot\max\limits_{1\leq i\leq n}|V_i|^2)$(其中 T 为最大迭代步数,N 为随机生成的测试数据个数,n 为待测软件因素的个数,$|V_i|$ 为因素 f_i 的取值).

以下我们通过实例说明如何应用交叉熵算法每次生成一条组合测试用例,直至生成覆盖所有因素的两两组合覆盖的测试用例集.

不妨设系统有 3 个因素:A,B,C,其中因素 A 有两个取值 A_1,A_2,因素 B 有两个取值 B_1,B_2,因素 C 有 3 个取值 C_1,C_2,C_3,可表示一个有向图,如图 2 所示.

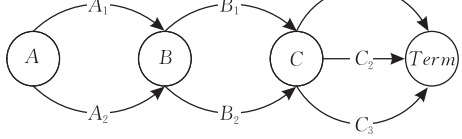

图 2 具有 3 个因素的有向图

其中节点 A,B,C 表示系统的因素,节点 *Term* 表示虚拟终端节点,每个节点出边为相应因素的取值.则从节点 A 到虚拟终端节点 *Term* 的一条有向路经为一条测试数据,例如($A_1\to B_2\to C_1$)或用三元组(A_1,B_2,C_1)表示.

首先初始化每个节点的各个有向出边的选择概率,这里取各个节点所有出边的等概率分布,即 $P_0(A_1)=P_0(A_2)=\frac{1}{2}$,$P_0(B_1)=P_0(B_2)=\frac{1}{2}$,$P_0(C_1)=P_0(C_2)=P_0(C_3)=\frac{1}{3}$;接着根据算法 2 的第 2 步:利用各条出边的选择概率,随机生成 N 条测试数据(不失一般性,设 $N=100$),例如(A_1,B_2,C_1),(A_1,B_2,C_2),…,(A_2,B_1,C_3),计算这 $N=100$ 条测试数据的目标函数适应值 $S_1,\cdots,S_{100}$(这里目标函数适应值定义为该测试数据覆盖所有

参数取值两两组合对中未被已生成的测试数据集 Ts 覆盖的对数)，根据目标函数适应值将这 $N=100$ 条测试数据排序，若有测试数据最优目标函数适应值恰好为 $C_n^2=\frac{n(n-1)}{2}=\frac{3\cdot(3-1)}{2}=3$，$n$ 为因素个数，这里 $n=3$，则将此测试数据放入覆盖集 Ts，回到步骤 1 生成下一条测试数据，否则选择目标函数适应值最好的前 $5(\lceil\rho N\rceil=\lceil 0.05\times 100\rceil=5,\rho=0.05)$ 条测试数据；由算法的第 3 步计算这前 5 条测试数据中每个因素各个取值选择概率，具体体现为计算这 $N=100$ 条测试数据中前 5 条测试数据每个因素各个取值出现的频率作为概率，不妨设为 $P_1'(A_1)=\frac{2}{5}$，$P_1'(A_2)=\frac{3}{5}$，$P_1'(B_1)=\frac{1}{5}$，$P_1'(B_2)=\frac{4}{5}$，$P_1'(C_1)=\frac{1}{5}$，$P_1'(C_2)=\frac{2}{5}$，$P_1'(C_3)=\frac{2}{5}$，应用平滑技术计算每个因素各个取值选择概率为 $P_1(A_1)=\alpha P_0(A_1)+(1-\alpha)P_1'(A_1)=\frac{9}{20}$，$P_1(A_2)=\frac{11}{20}$，$P_1(B_1)=\frac{7}{20}$，$P_1(B_2)=\frac{13}{20}$，$P_1(C_1)=\frac{4}{15}$，$P_1(C_2)=\frac{11}{30}$，$P_1(C_3)=\frac{11}{30}$；再重复上述两个步骤直到每个因素各个取值选择概率不再变化，或达到最大迭代步数，得到每个因素各个取值最优选择概率；由上述每个因素各个取值最优选择概率，再根据算法 3 生成 N 条测试数据，选出具有目标函数最优适应值的一条测试数据放入覆盖集 Ts 中，这里不妨设为 (A_1,B_1,C_3)；最后再重复前面步骤再生成一条最优测试数据放入覆盖集 Ts 中，直至覆盖系统中所有因素取值的两两组合. 这里最后生成三因素 A,B,C 所有因素取值的两两组合覆盖集为 $\{(A_1,B_1,C_3),(A_1,B_2,C_1),(A_1,B_2,C_2),(A_2,B_1,C_1),(A_2,B_1,C_2),(A_2,B_2,C_3)\}$.

4　基于粒子群算法的组合测试数据生成

粒子群优化算法(Particle Swarm Optimization，PSO)是由 Kennedy 和 Eberhart 等[20]于 1995 年提出的一种基于种群搜索的自适应演化计算技术. 在 PSO 算法中，用粒子的位置表示待优化问题的解，每个粒子性能的优劣程度取决于待优化问题目标函数确定的适应值，每个粒子由一个速度决定其飞行方向和速率大小. 设在一个 d 维的目标搜索空间中，有 M 个粒子组成一个群体，其中，在第 t 次迭代时粒子 i 的位置表示为 $X_i^t=(x_{i1}^t,x_{i2}^t,\cdots,x_{id}^t)$，相应的飞行速度表示为 $V_i^t=(v_{i1}^t,v_{i2}^t,\cdots,v_{id}^t)$. 在每一次迭代中，粒子通过跟踪两个极值来更新自己的速度和位置：一个极值是粒子本身迄今搜索到的最优解 $pBest$，称为个体极值，另一个极值是整个粒子群到目前为止找到的最优解 $gBest$，称为全局极值. 具体地讲，在第 $t+1$ 次迭代计算时，粒子 i 的速度和位置的更新规则分别为

$$v_{ij}^{t+1}=\omega v_{ij}^t+c_1r_1(pBest_{ij}^t-x_{ij}^t)+c_2r_2(gBest_j^t-x_{ij}^t) \tag{12}$$

$$x_{ij}^{t+1}=x_{ij}^t+v_{ij}^{t+1} \tag{13}$$

其中 ω 为惯性权重，c_1,c_2 为两个学习因子，r_1,r_2 为两个均匀分布在(0，1)间的随机数，$i=1,2,\cdots,M$；$j=1,2,\cdots,d$，本文采用文献[21]推荐，取参数 $c_1=c_2=1.49$，$\omega=0.729$.

粒子群优化算法是根据全体粒子和自身的搜索经验向着最优解的方向“飞行”，在进化过程中，开始优化速度较快，接近极小点时，优化速度极为缓慢. 如果一旦存在局部最优点，则一般很难突破局部最优点，这是不足之处. 为了刺激群体持续进化，避免群体的早熟收敛和停滞现象，在粒子群优化算法的基本框架上，我们结合迭代局部搜索的扰动机制[22]，利用其对当前解邻域内的局部详细搜索机制在解空间内不断变换解的位置以达到全局最优的扰动，增加粒子在局部详细搜索的能力同时增强其跳出局部最优的能力，从而避免陷入局部最优，得到问题的最优解或近优解. 下面我们把粒子群优化算法应用到组合测试数据生成上，这里把一条测试数据当作一个粒子，测试数据集当作粒子群，算法的基本流程与说明如算法 4 所示.

算法 4.　基于粒子群算法的组合测试数据生成.

1. 随机生成粒子群(即测试数据集)，为所有粒子位置(即各个参数取值)和速度赋初值；

2. 将粒子(即测试数据)的 $pBest$ 设置为当前位置，$gBest$ 设置为初始群体(即测试数据集)中最佳粒子的位置；

3. 对粒子群(即测试数据集)中的每个粒子 i(即每条测试数据)：由式(12)和式(13)更新当前粒子的度和位置 x_i^t；如果位置 x_i^t 不满足参数水平的约束则回飞到上一位置 x_i^{t-1}；

3.1. 根据某种扰动机制，对当前位置 x^t 进行扰动，生成新的可行解，并在其邻域内确定最优 x''；

3.2. 比较 x^t 和 x''，如果 x'' 优于 x^t，则令粒子 i 当前位置为 x''，否则令其位置为 x^t；

4. 更新当前粒子 i(即测试数据)的最优位置 $pBest_i$；

5. 更新种群(即测试数据集)的最优位置 $gBest$；

6. 重复步 3 直到终止条件满足.

输出：一条最优组合测试数据.

粒子表达式与粒子群的初始化. 用粒子 i 的每个位置 $x_{ij}(j=1,2,\cdots,n)$ 分别表示 n 个参数的具体取值，则每个粒子 i 可表示一条组合测试数据，即一条组合测试数据是由 $x_{ij}(x_{ij}\in V_j, j=1,2,\cdots,n)$ 组成的一个向量，其中 n 为因素个数. 采用随机生成的方法生成具有 M 个粒子(测试数据)的粒子群(测试数据集)，在每个粒子(测试数据) i 生成过程中，每个参数 f_j 的具体取值 x_{ij} 在 $\{V_{j1},V_{j2},\cdots,V_{jk}\}$ 中以等概率分布均匀取值，其中 k 为第 j 个参数 f_j 的水平数 $k=|V_j|$.

适应值计算. 粒子的适应值用来说明粒子所表示解的质量. 在两两覆盖组合测试数据的生成中，所有参数两两组合对中未被已生成的测试数据集 S 覆盖，而被新生成的一个组合测试数据所覆盖的对数为优化的目标函数，因此，适应值函数 $F(S)$ 通过查找所有参数两两组合对中未被 S 覆盖，而被新生成的组合测试数据所覆盖的对数来确定，最优解即为一个覆盖所有参数两两组合对中未被 S 覆盖的对数最大的组合测试数据.

参数水平约束的处理. 粒子群算法已经成功应用于求解带约束的优化问题，处理约束的最常用方法是使用惩罚函数，然而，试验表明这种方法降低 PSO 算法的效率. 这是因为当出现不可行粒子，则将粒子的位置重新设为粒子在此之前的最佳位置 $pBest$，这样将会阻止粒子朝着全局最优解搜索. 最近由 Li 等[23]引入一种称为“回飞机制”的约束处理方法，对于大多数带约束的优化问题，其全局最优解一般都位于或接近于可行区域的边界. 粒子在可行区域内初始化，当寻优过程开始时，粒子在可行空间内搜索解. 如果粒子飞入非可行解区域，即其取值超过参数的水平值时，它将被迫飞回到前一位置，以保证解是可行解. 粒子飞回的前一位置可能在下一次迭代中更接近于边界，这就使得粒子以更大的概率飞向全局最优解，试验结果表明，这种机制比其它约束处理方法更易搜索到最优解.

扰动机制. 对粒子新生成位置 x 的每一维以概率 $p=1/d$ 进行扰动，这样可增加粒子的组件随机性，具体实现为：对于粒子 i 的每一维 j，在 $[0,1]$ 中生成随机数 r，当 $r\leq 1/d$ 时，在 $\{V_{j1},V_{j2},\cdots,V_{jk}\}$ 中随机选择一个替换对应参数的具体取值 x_{ij}，其中 k 为第 j 个参数 f_j 的水平数.

算法复杂度为 $O(T\cdot N\cdot C_n^2\cdot \max\limits_{1\leq j\leq n}|V_j|^2)$(其中 T 为最大迭代步数，N 为随机生成的测试数据(粒子)个数，n 为待测软件因素的个数，$|V_j|$ 为因素 f_j 的取值).

下面我们应用前面的实例，用粒子群算法描述每次生成一条组合测试用例的过程，直至生成覆盖所有因素的两两组合覆盖的测试用例集：

首先随机生成 N 条测试数据集(粒子群)，这里不妨设 $N=100$ 例如 $(A_1,B_2,C_1),(A_1,B_2,C_2),\cdots,(A_2,B_1,C_3)$ 将其相应表示为 $(1,2,1),(1,2,2),\cdots,(2,1,3)$，并以此作为这 $N=100$ 条测试数据(粒子)初始位置. 速度取值为 $(-1,1)$ 上的均匀分布随机数，例如，测试数据 $(1,2,1)$ 的初始位置为 $x_0^0=(1,2,1)$，初始速度随机产生为 $v_0^0=(-0.2,0,0.1)$，测试数据 $(1,2,2)$ 的初始位置为 $x_1^0=(1,2,2)$，初始速度随机产生为 $v_1^0=(0.2,-0.3,-0.1)$；由算法第 2 步知：将这 $N=100$ 条测试数据的(粒子)当前位置设置为当前测试数据(粒子)的最佳位置 $pBest$，且计算这 $N=100$ 条测试数据(粒子)的目标函数适应值(这里目标函数适应值定义为该测试数据覆盖所有参数取值两两组合对中未被已生成的测试数据集 Ts 覆盖的对数)，找出具有最优目标函数适应值所对应的测试数据(粒子)位置，作为这 $N=100$ 条测试数据集(初始粒子群)最优位置 $gBest$，例如，这里不妨设测试数据 (A_1,B_2,C_2) 的目标函数适应值最优，即 $gBest=x_1^0=(1,2,2)$；若 $gBest$ 恰好为 $C_n^2=n(n-1)/2=3\cdot(3-1)/2=3$，$n$ 为因素个数，这里 $n=3$，则将此测试数据放入覆盖集 Ts，回到步 1 生成下一条测试数据. 第 3 步：将这 $N=100$ 条测试数据集中每条测试数据(粒子)由式(12)和(13)更新该测试数据的速度和位置，具体计算体现为：不妨取测试数据 (A_1,B_2,C_1) 其相应的位置为 $x_0^0=(1,2,1)$，速度为 $v_0^0=(-0.2,0,0.1)$，且 $pBest=x_0^0=(1,2,1)$，测试数据 (A_1,B_2,C_3) 的目标函数适应值作为这 $N=100$ 条测试数据集中最优测试数据，即 $gBest=x_1^0=(1,2,3)$，于是由

$$v_0^1=\omega v_0^0+c_1r_1(pBest-x_0^0)+c_2r_2(gBest-x_0^0),$$

$$x_0^1=x_0^0+v_0^1,$$

这里 $\omega=0.729$，$c_1=c_2=1.49$，$r_1=r_2=0.5$，故有

$$v_0^1=0.729\times(-0.2,0,0.1)+1.49\times0.5\times(x_0^0-x_0^0)+1.49\times0.5\times[(1,2,3)-(1,2,1)]=(-0.1458,2,1.5629),$$

$$x_0^1=x_0^0+v_0^1=(1,2,1)+(-0.1458,0,0.8179)=(0.8542,2,2.5629),$$

将上述 x_0^1 取整得 $x_0^1=(0,2,2)$；由于 $0\notin\{1,2\}$，故

利用回飞机制可得 $x_0^1=(1,2,2)$，再利用扰动机制(取值不变的数值再以相应该维取值的均匀分布进行变化)，可得 $x_0^1=(2,2,2)$；第 4 步：计算更新后这 $N=100$ 条测试数据(粒子)的目标函数适应值，根据目标函数适应值更新相应 $N=100$ 条测试数据(粒子)的 $pBest$(这里如果更新后测试数据的目标函数适应值优于与原来相应测试数据的目标函数适应值则更新 $pBest$ 否则相应的 $pBest$ 取值不变)，更新这 $N=100$ 条测试数据集(粒子群)的 $gBest$(若更新后的测试数据集的最优目标函数适应值优于原更新前的测试数据集的最优目标函数适应值则更新 $gBest$ 否则相应的 $gBest$ 取值不变)；重复前面第三步直到终止条件满足(即找到全局最优解或达到最大迭代步数). 最后输出具有最优解所对应的一条测试数据(粒子)放入测试覆盖集中，这里不妨设测试数据(A_1,B_2,C_3)为具有最优解所对应的测试数据；最后重复上述步骤再生成一条具有全局最优的测试数据放入覆盖集中，直至覆盖系统中所有因素取值的两两组合. 这里最后生成三因素 A,B,C 的所有因素取值的两两组合覆盖集为$\{(A_1,B_2,C_3),(A_1,B_2,C_2),(A_1,B_1,C_1),(A_2,B_1,C_2),(A_2,B_1,C_3),(A_2,B_2,C_1)\}$.

5　测试数据集的约简

由交叉熵方法和粒子群算法生成的两两组合覆盖测试数据集往往还有冗余，可以根据文献[24]对其作进一步的约简，这里我们给出相关定义及相应方法说明. 首先给出如下定义.

定义 3.　设 $t_1=(x_1,\cdots,x_n)$ 为待测软件 SUT 的测试数据集 A 中的一个测试数据，如果这个 n 元组在某个位置 i(第 i 个参数)的取值与其它 $n-1$ 个位置上取值形成的 $n-1$ 个组合对，已经出现在 $A\backslash\{t_1\}$ 的测试数据所形成组合对中，则称 t_1 的位置 i 相对于 $A\backslash\{t_1\}$ 为不关心位置，否则称为关心位置.

例如，$A=\{(1,2,3),(1,2,1),(1,1,3)\}$，取$t_1=(1,2,3)$则 t_1 中第一个位置为不关心位置，因为由第一位置所构成的组合对$(1,2,-),(1,-,3)$分别出现在$(1,2,1)$和$(1,1,3)$中.

由定义 3 给出如下测试数据集约简方法.

情形 1.　当一个测试数据 t_i 的所有位置相对于测试数据集 $Ts\backslash\{t_i\}$ 都是不关心位置时，此测试数据为冗余测试数据，删除后不会影响测试数据集的两两组合覆盖率.

情形 2.　当两个测试数据 t_i 和 t_j 的各个参数对应位置取值相同，或是为相对于测试数据集 $Ts\backslash\{t_i,t_j\}$ 的不关心位置，则这两个测试数据合并为一个测试数据不会影响整个测试数据集的两两组合覆盖率.

下面我们来说明情形 1 与 2 的正确性. 就情形 1 而言，由于测试数据 t_i 的所有位置都是不关心位置，故其每个位置的取值与其它位置的取值所形成组合对，都出现在 $Ts\backslash\{t_1\}$ 的测试数据所形成组合对中，从而测试数据集 Ts 的两两组合覆盖率与 $Ts\backslash\{t_1\}$ 两两组合覆盖率相同；对于情形 2，当两个测试数据 t_i 和 t_j 各个参数对应位置取值相同(即 $t_i=t_j$)，则这两个测试数据合并为一个测试数据时，不会影响整个测试数据集的两两覆盖率，如果 t_i 和 t_j 含有不关心位置，当 t_i 和 t_j 的不关心位置相同，则合并后该位置仍然是不关心位置，从而该位置取值所形成的两两组合覆盖率相同，若 t_i 和 t_j 的不关心位置不相同，则合并后相应位置取值为不关心位置的值，即 $t_i=(a_1,\cdots,a_{i-1},a_i,a_{i+1},\cdots a_{j-1},*,a_{j+1},\cdots,a_n)$，$t_j=(a_1,\cdots,a_{i-1},*,a_{i+1},\cdots a_{j-1},a_j,a_{j+1},\cdots,a_n)$，合并为 $t=(a_1,\cdots,a_{i-1},a_i,a_{i+1},\cdots,a_{j-1},a_j,a_{j+1},\cdots,a_n)$，则 t_i 的位置 a_i 与 t_j 的位置 a_j 的取值在 Ts 中两两组合覆盖率同合并后在 $Ts\backslash\{t_1,t_2\}\cup\{t\}$ 中两两组合覆盖率一样，这是因为不关心位置的取值 $*$ 不影响 a_i 与 a_j 在合并前后取值的两两组合覆盖率，从而情形 2 是正确的.

举例：不妨设有两条测试数据$(*,2,*,1,3,4)$、$(*,2,1,1,*,4)$(其中$*$表示为不关心位置)，根据情形 2 可合并为一条测试数据$(*,2,1,1,3,4)$.

根据以上给定的约简方法，我们给出如算法 5 所示的组合测试数据集约简算法.

算法 5.　组合测试数据集的约简.

输入：一个两两组合覆盖测试数据集 Ts

1. 从 Ts 中依次选出测试数据 t_i，标注 t_i 相对于 $Ts\backslash\{t_i\}$ 的不关心位置；如果测试数据 t_i 每个位置都是相对于 $Ts\backslash\{t_i\}$ 的不关心位置，根据情形 1 直接从测试数据集 Ts 中删除测试数据 t_i；

2. 从 Ts 中依次选择两个测试数据 t_i 和 t_j，标注 t_i 和 t_j 相对于 $Ts\backslash\{t_i,t_j\}$ 的不关心位置，由情形 2 判断是否能合并，若能则合并 t_i 和 t_j 为一个新的测试数据 t_c，加入到当前测试数据集 Ts，并从中删除 t_i 和 t_j.

输出：一个约简后规模更小的两两组合覆盖测试数据集.

举例：不妨设有一四因素系统(A,B,C,D)，其中因素 A 有 4 个取值，因素 B 有两个取值，因素 C

有两个取值,因素 D 有两个取值,现有覆盖所有因素两两组合的测试用例集:

$Ts=\{(1,2,2,2),(1,1,1,2),(1,2,2,1),(2,1,1,1),(2,2,2,2),(3,1,2,1),(3,1,1,2),(3,2,1,1),(4,1,1,1),(4,2,2,2)\}$.

下面应用约简算法对其进行约简:首先对测试数据集中每条测试数据标注不关心位置,找出是否有测试数据中每个位置都标注为不关心位置,根据算法这里只有第一条测试数据 $t_1=(1,2,2,2)$ 中的每个位置是不关心位置,而其余均不是,这是由于 $t_1=(1,2,2,2)$ 中的第 1 个因素取值 1 所构成的组合对(1,2,−,−),(1,−,2,−),(1,−,−,2)出现在 $Ts\backslash\{t_1\}$ 的测试数据 $t_2=(1,1,1,2)$ 与测试数据 $t_3=(1,2,2,1)$ 中,故将其标注为 *,即不关心位置,测试数据 t_1 中第 2 个因素取值 2 所构成的组合对(1,2,−,−),(−,2,2,−),(−,2,−,2)在 $Ts\backslash\{t_1\}$ 中的测试数据 $t_3=(1,2,2,1)$,$t_5=(2,2,2,2)$ 出现,第 2 个因素的位置标注为 * 不关心位置;同样对测试数据 t_1 中第 3 个因素取值 2 与第 4 个因素取值 2,所构成的组合对(1,−,2,−),(−,2,2,−),(−,−,2,2)与组合对(1,−,−,2),(−,2,−,2),(−,−,2,2)分别出现在 $Ts\backslash\{t_1\}$ 的测试数据 $t_3=(1,2,2,1)$,$t_5=(2,2,2,2)$ 与测试数据 $t_2=(1,1,1,2)$,$t_5=(2,2,2,2)$ 中,测试数据 t_1 的第 3 与第 4 位置也都标注为 * 不关心位置;从而测试数据 t_1 的所有位置都标注为 * 位置,根据情形 1 直接从测试数据集 Ts 中删除测试数据 t_1;接着在删除测试数据 t_1 后对覆盖集 $Ts'=Ts\backslash\{t_1\}$ 中依次选取两条测试数据标注其相应不关心位置,只有 $Ts'=Ts\backslash\{t_1\}$ 中的测试数据 $t_6=(3,1,2,1)$ 的第 4 位置标注为 * 不关心位置与 $t_7=(3,1,1,2)$ 的第 3 位置标注为 * 不关心位置;这是因为测试数据 $t_6=(3,1,2,1)$ 的第 4 位置 1 所构成的组合对(3,−,−,1),(−,1,−,1),(−,−,2,1)由 $Ts'\backslash\{t_6,t_7\}$ 中的测试数据 $t_3=(1,2,2,1)$,$t_8=(3,2,1,1)$,$t_9=(4,1,1,1)$ 覆盖,$t_7=(3,1,1,2)$ 的第 3 位置 1 所构成的组合对(3,−,1,−),(−,1,1,−),(−,−,1,2)由 $Ts'\backslash\{t_6,t_7\}$ 中的测试数据 $t_2=(1,1,1,2)$,$t_8=(3,2,1,1)$ 覆盖,故将 $t_6=(3,1,2,1)$,$t_7=(3,1,1,2)$ 根据情形 2 合并为新的测试数据 $t=(3,1,2,2)$ 放入 Ts' 中并删除 $t_6=(3,1,2,1)$ 与 $t_7=(3,1,1,2)$,形成约简后的覆盖集:

$Ts=\{(1,1,1,2),(1,2,2,1),(2,1,1,1),(2,2,2,2),(3,1,2,2),(3,2,1,1),(4,1,1,1),(4,2,2,2)\}$.

6 实验分析

为了检验基于交叉熵(CE)方法和粒子群算法(PSO)的组合测试数据生成方法的效果,我们分别实现了基于 CE 方法和 PSO 算法的测试数据自动生成工具,并进行了多次实验.本节将对实验数据进行分析,并以如下符号表示具体的 SUT:$P^Q\times U^V$ 表示 SUT 有 $Q+V$ 个参数,其中 Q 个参数每个有 P 个取值,V 个参数每个有 U 个取值.例如:$4^{12}\times 7^5$ 表示 SUT 有 17 个参数,其中 12 个参数有 4 个取值,其余 5 个参数每个有 7 个取值.实验中对于 CE 方法中参数 α 取为 0.4,而最大迭代数 $I_{\max}$ 取 20,PSO 算法中的参数粒子数取值为 1000,最大迭代数 $I_{\max}$ 取值为 10.

由于 CE 方法和 PSO 算法都是启发式方法,不能保证在任何情况下都能生成最小两两组合覆盖表.为了检查它们生成测试数据的效果,我们将从 3 个方面考虑:首先是将 CE 方法和 PSO 算法与其它同类方法在生成规模上进行比较,其次是将 CE 方法和 PSO 算法与其它常见演化方法在时间上给予比较,最后是 CE 方法和 PSO 算法的自身相关参数与最大迭代次数对生成测试数据规模的影响.

实验 1. 测试数据集规模比较.

我们将 CE 方法和 PSO 算法与其它同类方法在生成规模上进行比较.实验中比较的对象有 AETG 系统、PAIRTEST 系统、Kobayashi 等提出代数方法、Williams 提出的代数方法、基于网络图组合测试工具 NetWork[11],基于解空间树的组合测试工具[13]、SA(模拟退火)方法[24]、GA(遗传算法)方法[24]、ACA(蚁群算法)方法[24],如表 1、表 2 所示;表 1 前 6 种方法的实验数据与表 2 前 4 种算法的实验数据均来自公开文献[13].而表 1 中的 SA、GA 和 ACA 方法的实验数据来自公开文献[24].

由表 1、表 2 可知 CE 方法和 PSO 算法生成的测试数据与其它方法生成的测试数据数量相当,在某些情况下生成的测试规模要小(例如表 2 中类型 4^6 与 7^9 等);另外由表 1 表明 CE 方法和 PSO 算法生成的测试数据规模总体相当,而由表 2 表明以数理统计为基础的 CE 方法总体上要比以仿生学为基础的 PSO 算法生成的测试数据规模要好,这与 CE 方法有更好的数学优化背景有关,而 PSO 算法是模拟生物种群群体智能优化方法,其结果是大部分近优.

表 1　测试数据集规模比较

算法	测试数据个数					
	(3^4)	(3^{13})	(2^{100})	$(3^{12}\times4^5)$	$(4^1\times3^{39}\times2^{35})$	$(5^3\times4^4\times3^1\times2^2)$
AETG[13]	9	15	10	31	28	31
PAIRTEST[13]	9	19	15	29	29	37
Williams[13]	9	17	14	28	40	45
Kobayashi[13]	9	15	16	28	36	50
NetWork[13]	9	19	70	26	49	34
PSST[13]	9	20	16	28	30	33
SA[24]	9	16	NA	NA	21	NA
GA[24]	9	17	12	NA	27	NA
ACA[24]	9	17	13	NA	27	NA
CE	9	17	13	28	30	34
PSO	9	17	13	29	32	33

注：NA 表示文献[24]中没有给出相应的有效数据.

表 2　对于 p^n（p 为素数或素数方幂）型测试数据生成规模比较

算法	测试数据个数								
	(4^5)	(4^6)	(7^8)	(7^9)	(8^9)	(8^{10})	(11^{10})	(11^{12})	(11^{13})
AETG[13]	28	28	80	87	116	121	207	221	229
PAIRTEST[13]	16	28	85	91	132	136	236	260	275
NetWork[13]	24	24	49	91	140	145	121	231	231
PSST[13]	16	27	49	86	64	114	121	121	230
CE	19	22	74	81	106	113	210	235	252
PSO	19	22	78	82	111	125	238	261	271

实验 2.　算法时间效率比较.

我们将 CE 方法和 PSO 算法与其它常见演化方法在时间上给予比较，采用的是同 SA（模拟退火）方法、GA（遗传算法）方法、ACA（蚁群算法）方法在时间上给与比较，详见表 3；表 3 中的 SA、GA 和 ACA 方法的时间数据来自公开文献[24].

表 3　运行时间比较

算法	运行时间/s					
	10^{10}	10^{20}	$5^1 3^8 2^2$	$7^1 6^1 5^1 4^5 3^8 2^3$	$5^1 4^4 3^{11} 2^5$	$6^1 5^1 4^6 3^8 2^3$
SA[24]①	NA	10833	214	874	379	579
GA[24]②	886	6365	22	207	124	149
ACA[24]②	1180	7083	31	258	154	188
CE③	985	4378	91	689	593	556
PSO③	244	1801	35	301	202	206

注：① C++，Linux，INTEL Pentium IV 1.8GHz；② C，Windows XP，INTEL Pentium IV 2.26GHz；③ Matlab，Windows XP，INTEL Core (TM)$_2$ 2.66GHz.

由表 3 知 CE 和 PSO 在某些情况下其时间效率上比 GA、ACA 要好，有些情况下差不多，有些情况要比 GA、ACA 差一些，在整体上比 SA 效率都好，表明 SA 在时间效率上最差，CE 和 PSO 与 GA、ACA 在整体上效率相当；就 CE 和 PSO 相比，PSO 在时间效率上要比 CE 整体上都好，PSO 比 CE 在时间上收敛速度快.

实验 3.　算法参数分析.

为了分析 CE 方法和 PSO 算法的自身相关参数与最大迭代次数对生成测试数据规模的影响，我们采用 SUT 类型是 (4^6) 型与 $(5^3\times4^4\times3^1\times2^2)$ 型作为实例，在其它参数不变的情形下，变化其中一个主要参数（如 CE 的参数 α，PSO 的粒子数以及最大迭代次数 $I_{\max}$），实验在相同条件下进行 50 次，用统计学中的盒图（又称箱线图）分析比较参数变化对生成测试数据（这里数据没有约简）规模的影响. 盒图的最下端一横表示数据中的最小值，最上端一横表示数据中的最大值，盒图的中间一横表示中位数取值，盒子的大小表示数据的离散程度. 图 3 和图 4 分别给出了两种不同 SUT 类型下，CE 方法中参数参数 α 变化时对生成测试数据集规模大小的影响，图 5，图 6 给出了 CE 方法中最大迭代数 $I_{\max}$ 变化对生成测试数据规模的影响.

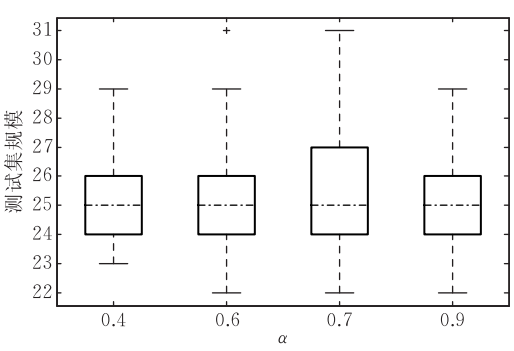

图 3 4^6型测试集规模

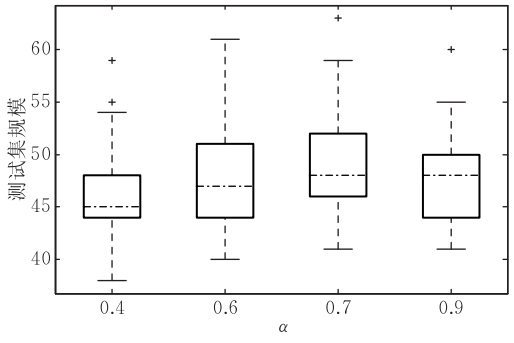

图 4 $5^3\times4^4\times3^1\times2^2$型测试集规模

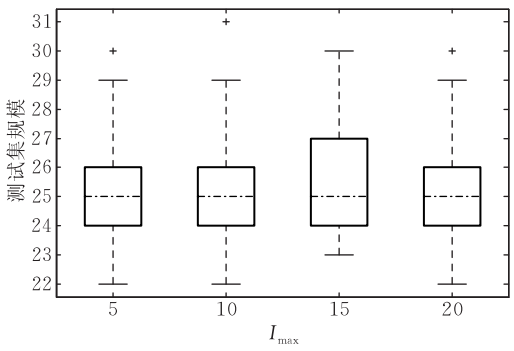

图 5 4^6型测试集规模

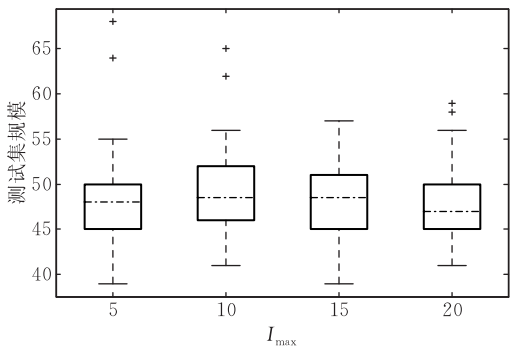

图 6 $5^3\times4^4\times3^1\times2^2$型测试集规模

由图 3 知 CE 方法的参数 α 四个取值所对应的盒图对于 4^6 型来说它们的中位数取值大小是相同的，α 取值为 0.4 时其所对应盒图的最小值较其余最小值大，而 α 取值为 0.7 所对应盒图较大，故其所产生测试数据离散程度较大. 由图 4 知 CE 方法的参数 α 四个取值所对应的盒图对于 $5^3\times4^4\times3^1\times2^2$ 型来说，α 取值为 0.4 其盒图的中位数取值最小，盒图的大小及最大与最小取值又是最小. 因而 α 取值为 0.4 所产生测试数据效果最好. 由图 5 和图 6 知 CE 方法的最大迭代数 I_{max} 取值 20 效果较好，因其盒图较小即数据较为集中并且在图 6 中其中位数取值最小.

图 7～图 10 为 PSO 方法中粒子数与最大迭代数 I_{max} 的变化对生成测试数据规模的影响.

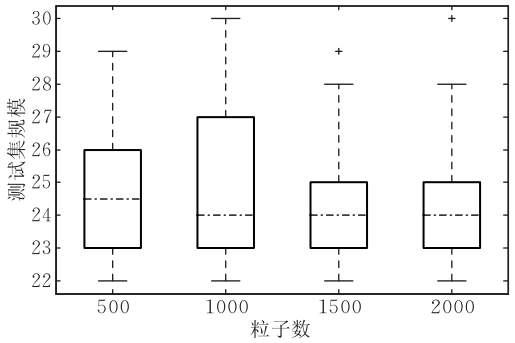

图 7 4^6型测试集规模

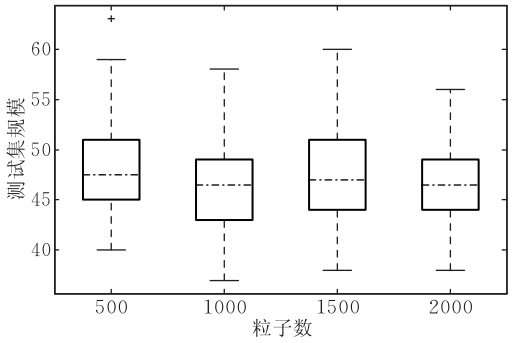

图 8 $5^3\times4^4\times3^1\times2^2$型测试集规模

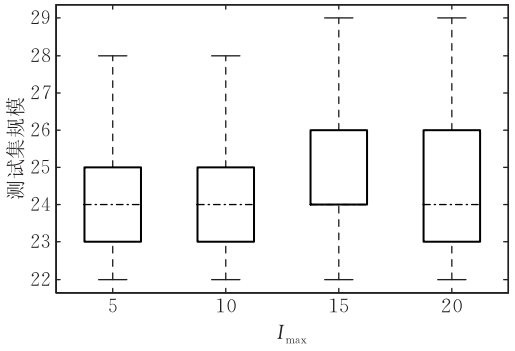

图 9 4^6型测试集规模

由图 7～图 8 知 PSO 方法中的粒子数取值为 1000 时其盒图中位数取值与最小值取值最小，不过其在图 7 中的盒图较大即数据离散度较大. 由图 9 和图 10 知 PSO 方法中的最大迭代数 I_{max} 取值为 10

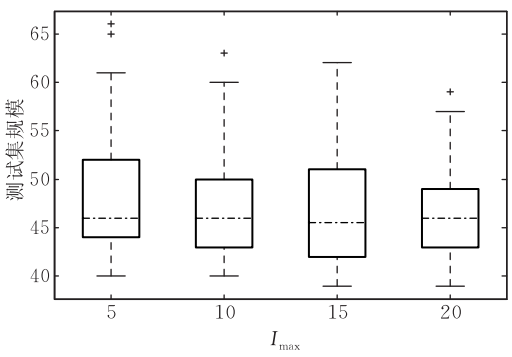

图 10　$5^3\times4^4\times3^1\times2^2$型测试集规模

较好，因其盒图中位数取值与其它取值差不多，且其盒图大小在两个图中变化不是很大，其它取值在两个图中盒图变化较大.

由上面实验分析知，这些启发式搜索方法中并不存在一个“最优”算法，在所有情况下都能生成最小测试数据集.这表明交叉熵方法和粒子群算法是有效的，可以作为已有方法的重要补充.

7　结论及进一步工作

本文提出以数理统计为基础的交叉熵方法和以仿生学为基础的粒子群算法来生成两两组合测试数据.交叉熵方法采用最优选择概率产生测试数据，而粒子群算法应用扰动机制，选择出一组规模较小的测试数据集，并对由两种算法所生成的测试数据集进行约简.实验表明交叉熵方法和粒子群算法是有效的，可以作为已有方法的重要补充.

关于组合测试的研究尽管已经取得了丰富的成果，但还存在很多问题：首先由于 $N(N\geq2)$维组合覆盖表以及多重维数组合覆盖表的构造问题是一个NP-hard 问题，现有算法大多为近似算法，算法的性能往往无法满足需求；其次，现有算法大多针对两两组合覆盖，对于多重维数组合覆盖的研究相对较少，而在很多情况下为提高软件可靠性，需要采取$N(N>2)$维组合覆盖和多重维数的组合测试，因此有必要对多重维数组合覆盖作进一步研究.

致　谢　衷心感谢南京大学软件新技术国家重点实验室的许蕾副教授及匿名审稿人对文章提出的宝贵意见，使得文章质量得到进一步的提高！

参　考　文　献

[1] Dunietz I S, Ehrlich W K, Szablak B D, Mallows C L, Lannino A. Applying design of experiments to software testing: Experience report//Proceedings of the 19th International Conference on Software Engineering. Boston, Massachusetts, USA, 1997: 205-215

[2] Kuhn D R, Reilly M J. An investigation of the applicability of design of experiments to software testing//Proceedings of the 27th NASA/IEEE Software Engineering Workshop. NASA Goddard Space Flight Center, 2002: 91-95

[3] Lei Y, Tai K C. In_Parameter_Oder: A test generation strategy for pairwise testing. Department of Computer Science, North Carolina State University, Raleigh, North Carolina: Technical Report TR-2001-03, 2001

[4] Mandl R. Orthogonal latin squares: An application of experimental design in compiler testing. Communications of the ACM, 1985, 28(10): 1054-1058

[5] Kobayashi N, Tsuchiya T, Kikuno T. A new method for constructing pair-wise covering designs for software testing. Information Processing Letters, 2002, 81(2): 85-91

[6] Williams A W. Software component interaction testing: Coverage measurement and generation of configurations [Ph. D. dissertation]. Ottawa-Carleton Institute for Computer Science, School of Information Technology and Engineering, University of Ottawa, Canada, 2002

[7] Cohen D M, Dalal S R, Fredman M L, Patton G C. The AETG system: An approach to testing based on combinatorial design. IEEE Transactions on Software Engineering, 1997, 23(7): 437-444

[8] Tung Y W, Aldiwan W S. Automating test case generation for the new generation mission software system//Proceedings of the IEEE Aerospace Conference. Big Sky, MT, USA, 2000: 431-437

[9] Yan Jun, Zhang Jian. Backtracking algorithms and search heuristics to generate test suites for combinatorial testing//Proceedings of the 30th Annual International Conference on Computer Software and Applications (COMPSAC06). Chicago, IL, 2006, 1: 385-394

[10] Cheng Xiang, Gu Qing, Li Ang, Cheng Daoxu. Variable strength interaction testing with an ant colony system approach//Proceedings of the 16th Asia-Pacific Software Engineering Conference. Penang, 2009: 160-167

[11] Nie Chang-Hai, Xu Bao-Wen. An automatic black-box test case generation algorithm based on interface parameters. Chinese Journal of Computers, 2004, 27(3): 382-388(in Chinese)
(聂长海，徐宝文. 基于接口参数的黑箱测试用例自动生成算法. 计算机学报，2004，27(3)：382-388)

[12] Nie Chang-Hai, Xu Bao-Wen, Shi Liang, Dong Guo-Wei. Automatic test generation for N-way combinatorial testing//Proceedings of the 2nd International Workshop on Software Quality (SOQUA 2005). Fair and Convention Center, Erfurt, Germany, 2005: 203-211

[13] Shi Liang, Nie Chang-Hai, Xu Bao-Wen. Pairwise test data generation based on solution space tree. Chinese Journal of Computers, 2006, 29(6): 849-857(in Chinese)

(史亮，聂长海，徐宝文. 基于解空间树的组合测试数据生成. 计算机学报，2006，29(6)：849-857)

[14] Nie Chang-Hai, Xu Bao-Wen, Shi Liang. A new pairwise covering test data generation algorithm for the system with many 2-level factors. Chinese Journal of Computers, 2006, 29(6): 841-848(in Chinese)
(聂长海，徐宝文，史亮. 一种新的二水平多因素系统两两组合覆盖测试数据生成算法. 计算机学报，2006，29(6)：841-848)

[15] Wang Zi-Yuan, Nie Chang-Hai, Xu Bao-Wen, Shi Liang. Optimal test suite generation methods for neighbor factors combinatorial testing. Chinese Journal of Computers, 2007, 30(2): 200-211(in Chinese)
(王子元，聂长海，徐宝文，史亮. 相邻因素组合测试用例集的最优生成方法. 计算机学报，2007，30(2)：200-211)

[16] Rubinstein R Y. The Cross-entropy method for combinatorial and continuous optimization. Methodology and Computing in Applied Probability, 1999, 1(2): 127-190

[17] Zhang De-Ping, Nie Chang-Hai, Xu Bao-Wen. Cross-entropy method based on Markov decision process for optimal software testing. Journal of Software, 2008, 19(10): 2770-2779 (in Chinese)
(张德平，聂长海，徐宝文. 基于 Markov 决策过程用交叉熵方法优化软件测试. 软件学报，2008，19(10)：2770-2779)

[18] De Boer P T, Kroese D P, Mannor S, Rubinstein R Y. A tutorial on the cross-entropy method. Annals of Operations Research, 2005, 134(1): 19-67

[19] Margolin L. On the convergence of the cross-entropy method. Annals of Operations Research, 2005, 134(1): 201-214

[20] Kennedy J, Eberhart R. Particle swarm optimization//Proceedings of the IEEE Conference on Neural Networks. Perth, Australia, 1995, 4: 1942-1948

[21] Kler M, Kennedy J. The particle swarm-Explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computer, 2002, 6(1): 58-73

[22] Cowlinga P I, Keuthenb R. Embedded local search approaches for routing optimization. Computers & Operations Research, 2005, 32(3): 465-490

[23] Li L J, Huang Z B, Liu F, Wu Q H. A heuristic particle swarm optimizer for optimization of pin connected structures. Computers and Structures, 2007, 85(4): 340-349

[24] Shiba T, Tsuchiya T, Kikuno T. Using artificial life techniques to generate test cases for combinatorial testing//Proceedings of the 28th Annual International Computer Software and Applications Conference(COMPSAC'04). Hong Kong, China, 2004, 1: 72-77

ZHA Ri-Jun, born in 1971, Ph. D. candidate, lecturer. His research interests include software testing and statistical testing etc.

ZHANG De-Ping, born in 1973, Ph. D., lecturer. His research interests include teaching and research on software testing, statistical testing and mathematical statistics.

NIE Chang-Hai, born in 1971, Ph. D., associate professor. His research interests include software engineering, software testing and fussy information processing etc.

XU Bao-Wen, born in 1961, Ph. D., professor. His research interests include teaching and research on programming language, software engineering, parallel and network, acquisition technique on knowledge and information etc.

Background

This paper is supported by the National Natural Science Foundation of China under grant Nos. 90818027, 60721002, 60873050, 60773104, the National High Technology Research and Development Program (863 Program) under grant Nos. 2008AA01Z143, 2009AA01Z147, the National Basic Research Program of China (973 Program) under grant No. 2009CB320703, and the Key Laboratory Funding of Science and Technology Commission of Shanghai Municipality under grant No. 09DZ2272600. These projects mainly research on the follows fields: Combinatorial coverage method and its effectiveness for software testing, the existence and the generation algorithm for several minimal combinatorial covering table, some techniques for software fault diagnosis and debugging based on combinatorial testing, the application of the combinatorial testing in Web testing and software configuration testing. The authors have made some works on the test case generation algorithm and the applications. This paper focuses on the research of the test data generation algorithm of combinatorial testing.