Software Engineering and Applications 软件工程与应用, 2020, 9(2), 148-158
Published Online April 2020 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2020.92018

Hans汉斯

Combinatorial Test Case Generation Method Based on Improved Particle Swarm Optimization Algorithm

Na Zhang[1], Yuting Jin[1,2], Xiaomei Tu[1,2], Liangliang Dong[1], Xiaoan Bao[1]

[1]School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou Zhejiang
[2]Zhejiang GuangSha College of Applied Construction Technology, Dongyang Zhejiang
Email: 1552474482@qq.com

Received: Apr. 9th, 2020; accepted: Apr. 22nd, 2020; published: Apr. 29th, 2020

Abstract

The combined test is difficult to achieve fully coverage about the all possibilities. In order to generate the least number of test cases, a method of test case set generation based on the combination of IPO strategy and CS-SPSO algorithm is proposed. Firstly, considering the characteristics of test cases to code particle, the initial population was divided into several subpopulations. Secondly, the simplified particle swarm optimization algorithm was used to search the optimal test cases in each subpopulation and communicated randomly among the populations. Then, the global search ability of cuckoo search was used to find the test case with better adaptability, and the dynamic step size was introduced to accelerate the convergence of the algorithm. Finally, combined it with the improved IPO strategy to generate combinational test cases. The experimental results show that, compared with the basic PSO, CS algorithm and the basic IPO Strategy, the proposed algorithm has certain advantages in case size and execution time.

Keywords

Combination Testing, Particle Swarm Optimization, Cuckoo Search, IPO Strategy

基于改进粒子群算法的组合测试用例生成方法

张　娜[1]，金瑜婷[1,2]，涂小妹[1,2]，董亮亮[1]，包晓安[1]

[1]浙江理工大学信息学院，浙江　杭州
[2]浙江广厦建设职业技术学院，浙江　东阳
Email: 1552474482@qq.com

收稿日期：2020年4月9日；录用日期：2020年4月22日；发布日期：2020年4月29日

文章引用: 张娜, 金瑜婷, 涂小妹, 董亮亮, 包晓安. 基于改进粒子群算法的组合测试用例生成方法[J]. 软件工程与应用, 2020, 9(2): 148-158. DOI: 10.12677/sea.2020.92018

摘　要

组合测试难以对所有可能取值的组合进行全面覆盖，针对如何生成最小测试用例集的问题，提出了一种基于约束处理和评估指标的类IPO策略和CS-SPSO算法相结合的测试用例集生成方法。首先，考虑测试用例的特性对粒子进行编码，将得到的初始种群划分为若干子种群。其次，利用简化粒子群算法搜索各子种群中的最优测试用例并在种群间进行随机交流。接着，利用布谷鸟搜索的全局搜索能力来寻找适应度更好的测试用例，引入动态步长加速算法收敛。最后，与改进的类IPO策略相结合生成组合测试用例集。实验结果表明，与基本的PSO、CS算法以及基本的IPO策略相比，本文提出的算法在用例规模和执行时间上具有一定的优势。

关键词

组合测试，粒子群算法，布谷鸟搜索，IPO策略

1. 引言

软件测试是保证软件质量的一个必不可少的环节[1]。由于人们对软件功能的要求越来越高，结构越来越复杂，影响软件质量的因素也变得越来越多。这些因素本身以及相互之间都有可能触发软件故障，对这些可能的影响因素进行百分百覆盖极耗费人力物力，因此选择少而精的测试用例就显得至关重要了[2]。组合测试作为一种基于规约的测试用例生产技术，能在保证错误检出率的前提下采用较少的测试用例来测试系统[3]。组合测试生成最小测试用例集是一个 NP 完全问题[4]，因此不少学者利用启发式搜索算法将其转换成搜索问题，进而生成最小的测试用例集[5]。例如 Sangeeta [6]、戚荣志[7]的并行化遗传算法；曾梦凡[8]、王小银[9]的蚁群算法；V. Chandra [10]、Bao [11]的粒子群优化算法等。

其中，粒子群优化算法(Particle Swarm Optimization, PSO)是 Kenney 和 Eberhart [12]提出的一种基于群体智能协作的全局随机搜索算法，相对于无法解决复杂优化问题的传统优化算法，粒子群算法以需调整的参数少、易实现、搜索与收敛快等优点引起了学术界的重视，广泛地应用在了组合测试用例生成中。粒子群算法用若干个测试用例作为初始化种群，种群中的一个粒子代表一种可能的测试用例；然后对粒子进行速度和位置更新，根据适应度函数找到个体最优解和全局最优解。

粒子群算法在生成组合测试用例上存在易陷入局部最优、收敛精度低、不易收敛等问题。因此，Chen [13]等人研究了搜索空间、适应值函数和启发式的合理设定提出了一种基于粒子群优化的成对组合测试用例集生成算法框架来提高单个测试用例的新组合覆盖能力。Yang [14]等人提出了以种群粒子优劣为依据对惯性权重进行自适应调整的方法来优化粒子群算法并将其与 one-test-at-time 策略相结合来生成组合测试用例集，该方法可解决粒子群算法易受配置参数影响的问题。Bao [11]等人提出了以粒子群早熟收敛程度为依据来动态调整惯性权值，将动态调整的简化粒子群优化算法应用到组合测试中提升了算法的基本检错能力和执行能力。Mahmoud [15]等人提出了一种基于模糊逻辑的自适应群优化算法，在生成覆盖数组大小时动态调整参数来克服优化过程中的问题并提高运算效率。但仍存在算法执行时间长，后期种群多样性差，组合因素选取的随机性等问题。

为了解决算法执行时间长、种群多样性差，组合测试的 NP 问题以及因素选取的随机性问题，本文提出一种基于改进的混合简化粒子群和布谷鸟优化算法(CS-SPSO)的组合测试用例集生成方法。该方法首先排除了速度这一不必要因素对粒子群优化过程的影响，加快了算法的运算速度；其次在初始化粒子群后，将其划分成 N 个子种群进行初步搜索，并在迭代过程中随机交流子种群间的当前最优粒子，增加了种群解的多样性；接着将 N 个子种群的最优解作为布谷鸟搜索的初始种群，避免了种群随机初始化导致的低离散度问题；同时在位置搜索中引入动态步长，解决种群寻优后期震荡，收敛速度慢的问题；最后使用基于约束处理和评估指标的类 IPO 策略生成组合测试用例集，不仅降低了排序的随机性，还提高了策略运行效率。

2. 基于粒子群和布谷鸟搜索的改进算法

2.1. 简化粒子群算法

粒子群算法是进化算法的一种，通过不断迭代并使用适应度函数来找到最优解。其改进研究有很多，其中 Li 等人[16]研究发现，粒子群算法可以没有粒子自身速度的概念，没有自身速度因素可避免人为确定参数最大值最小值范围对粒子收敛速度和精度的影响。从而提出了一种简化粒子群优化算法(Simplified Particle Swarm Optimization, SPSO)，具体定义如下：

假设 d 维的搜索空间中有 n 个粒子数，则第 i 个粒子在第 t 代的位置 $x_i=\left(x_{i1}^t,x_{i2}^t,\cdots,x_{id}^t\right)$，若个体历史最优位置为 $p_i=\left(p_{i1},p_{i2},\cdots,p_{id}\right)$，整个粒子群的历史最优解位置为 $g=\left(g_1,g_2,\cdots,g_d\right)$，则在第 $t+1$ 代时，第 i 个粒子在第 j 维空间中不含速度项的简化粒子群算法公式为：

$$x_{ij}^{t+1}=wx_{ij}^t+c_1r_1\left(p_{ij}^t-x_{ij}^t\right)+c_2r_2\left(g^t-x_{ij}^t\right) \tag{1}$$

其中，式(1)右边的第一项为历史部分，表示过去位置对现在所处位置的影响，其他两项的含义与基本粒子群算法中一样。

2.2. 布谷鸟搜索

布谷鸟搜索(Cuckoo Search, CS)是由剑桥大学 Yang 教授和 S. Deb [17]受自然界启发而提出的一种将布谷鸟育雏行为和 *Lévy* 飞行相互结合的全局搜索算法，具有全局搜索能力强、参数少、易实现等优点，但同时也存在收敛速度慢、进化后期种群多样性差等不足[18]。

假设 d 维的搜索空间中有 n 个鸟窝，则第 i 个鸟窝在第 t 代的位置为 x_i^t，布谷鸟搜索算法的搜索路径使用了随机性较强的 *Lévy* 飞行的搜索方式，那么布谷鸟寻找鸟窝的路径以及位置的更新公式为[19]：

$$x_i^{t+1}=x_i^t+\varepsilon\otimes L\left(\lambda\right) \tag{2}$$

其中，ε 为步长控制量；$\otimes$ 为点乘运算；$L\left(\lambda\right)$ 为莱维随机步长，λ 取值为(1, 3]。

宿主鸟以 Pa 概率发现外来鸟卵，用随机数 $r\in\left[0,1\right]$ 与概率 Pa 进行对比，如果 $r>Pa$ 就改变位置，反之不变，并结合适应度值可判断最优鸟窝。新建鸟窝位置的公式如下定义：

$$x_i^{t+1}=x_i^t+r\otimes Heaviside\left(Pa-\theta\right)\otimes\left(x_j^t-x_k^t\right) \tag{3}$$

其中，θ 为服从均匀分布的随机数；$Heaviside\left(x\right)$ 为跳跃函数，当自变量 $x>0$ 时，$Heaviside\left(x\right)=1$，反之为 0；$x_j^t$，$x_k^t$ 为第 t 代中的两个随机鸟窝位置。

2.3. CS-SPSO 算法

本文根据测试用例特征进行粒子编码，将简化粒子群算法的初始种群划分成若干子种群，在种群间

进行随机交流以提高搜索效率，基于各子种群的最优个体得到初始鸟窝位置，使用布谷鸟搜索进一步寻找全局最优个体，在搜索路径中增加了动态步长因子，使得最终生成的测试用例兼具收敛性和多样性。

基于各因素取值随机生成初始种群，种群中的粒子包含 d 个因素，即为一条可能的测试用例。首先将初始种群划分为 N 个子种群 $X=\{X_1,X_2,\cdots,X_N\}$，每个子种群有 n 个粒子 $X_i=\left(x_1^t,x_2^t,\cdots,x_n^t\right)$；当个体迭代更新时需根据其适应度函数值 $F\left(x_i\right)$ 进行评估，公式如下：

$$F\left(x_i\right)=Cov\left(x_i\right)/AllCov\left(x_i\right) \tag{4}$$

其中，$AllCov\left(x_i\right)$ 为需覆盖组合集的总个数，$Cov\left(x_i\right)$ 为测试用例覆盖需覆盖组合集的个数。

接着在子种群中利用简化粒子群算法进行最优值搜索，并根据式(1)进行粒子更新；在迭代过程中，每达到种群交流周期 t' 时，当前种群中的最优粒子根据式(5)进行随机交换。

$$xbest=\begin{cases}xbest_k, & fitness\left(xbest_k\right)>fitness\left(xbest\right)\\ xbest, & other\end{cases} \tag{5}$$

其中，$xbest$ 为当前种群最优解；$xbest_k$ 为第 k 个子种群的最优解，k 为区间 $[1,N]$ 上的随机整数。

迭代完成后，得到所有子种群的最优解集合 $X_{best}=\{xbest_1,xbest_2,\cdots,xbest_N\}$，根据式(2)对 X_{best} 进行多次更新并保留原解，当种群达到一定规模后作为布谷鸟搜索算法的初始种群。根据固定步长进行鸟窝位置更新易导致算法后期震荡，收敛时间变长，本文使用基于动态步长的 *Lévy* 飞行对当前种群更新，动态步长 ε 的公式如下：

$$\varepsilon=\varepsilon_{\min}+\left(\varepsilon_{\max}-\varepsilon_{\min}\right)\left(t_{\max}-t\right)/t_{\max} \tag{6}$$

其中，$\varepsilon_{\max}$ 和 $\varepsilon_{\min}$ 分别为步长的最大值和最小值，$t_{\max}$ 为最大迭代次数，t 为当前迭代次数。

位置更新后根据公式(3)对劣解进行更新，当达到最大迭代次数或满足停止条件时，输出最优解。

3. 改进的组合测试用例集生成

3.1. 基于 CS-SPSO 的单条测试用例生成

粒子群算法和布谷鸟搜索算法都是采用适应度函数值来衡量位置的优劣情况。在使用 CS-SPSO 算法生成测试用例时，粒子可能会飞出有效的搜索空间，为解决这个问题需进行边界处理。根据文献[14]的建议，本文将采用 Robinson 提出的反射墙策略，即当超过边界时使用下式对其进行反弹，反射墙公式如下：

$$f\left(x_{i,j}\right)=\begin{cases}2l_i-x_{i,j}+1, & \text{if } x_{i,j}>l_i\\ -x_{i,j}+1, & \text{if } x_{i,j}<1\\ x_{i,j}, & \text{else}\end{cases} \tag{7}$$

其中，l_i 代表初始粒子个数，结合以上策略，图 1 为基于 CS-SPSO 算法生成单条测试用例的流程图。

CS-SPSO 算法的时间复杂度由目标函数的时间复杂度 $F(n)$计算决定，当 $F(n)$运算阶数高于 n，时间复杂度为 $O(F(n))$；当相等或低于时，时间复杂度为 $O(n)$。根据本文测试用例生成算法的流程图 1 可知，种群划分增加了循环次数，但子种群数量较少并不会影响时间复杂度；在每一代引入动态步长因子也未增加时间复杂度。总的时间复杂度公式如下：

$$T\left(n\right)=O\left(F\left(n\right)\right) \tag{8}$$

算法的空间复杂度取决于种群规模 $N\cdot n$ 和搜索维度 D 的影响，计算公式如下：

$$S\left(n\right)=O\left(D\cdot N\cdot n\right) \tag{9}$$

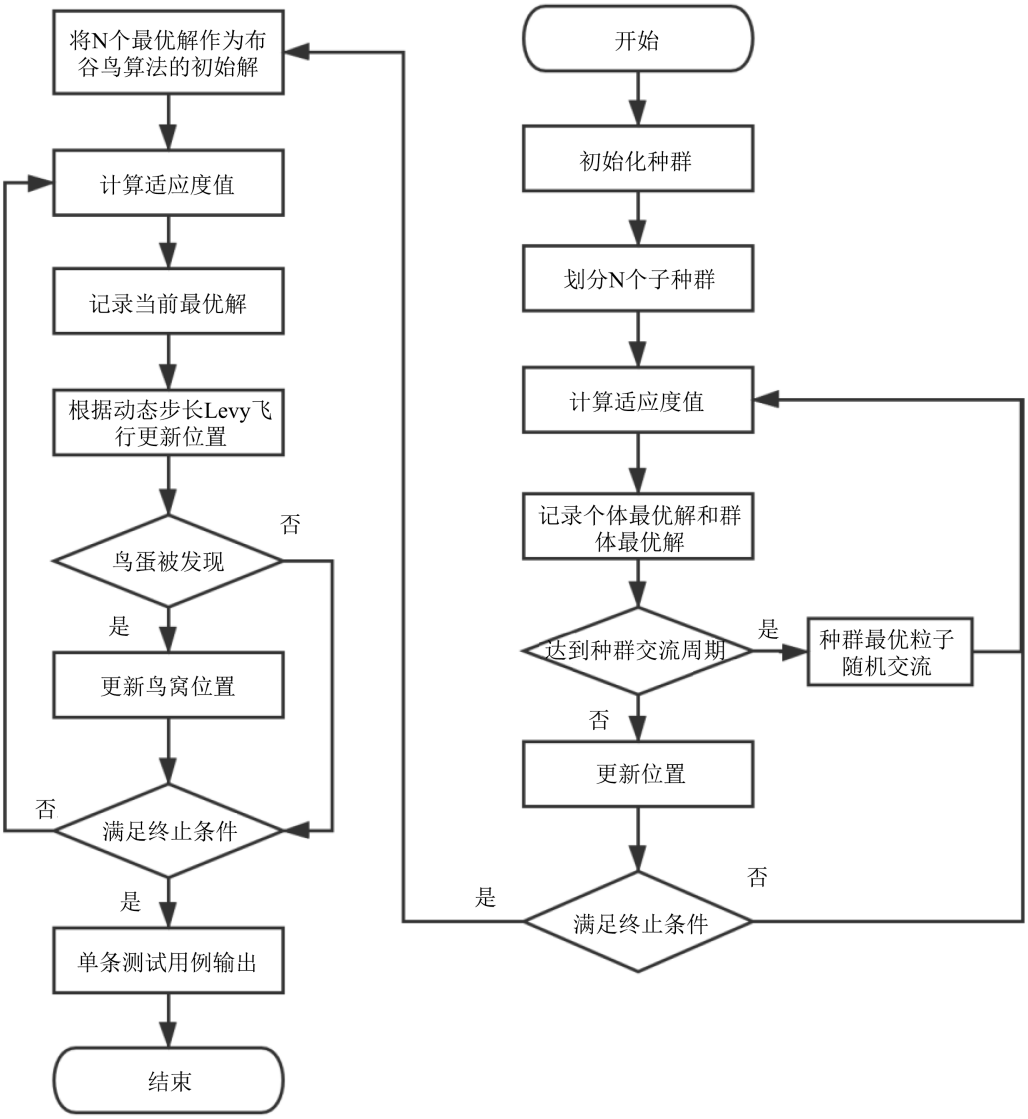


Figure 1. Flow chart of test case generation based on CS-SPSO
图 1. 基于 CS-SPSO 的测试用例生成流程图

本文所提负载均衡算法的算法复杂度随着维度的升高，其时间复杂度和空间复杂度都升高。

3.2. 改进的组合测试用例集生成方法

Tai 和 Lei [20]提出了一种不同于 one-test-at-time 的 IPO 策略，该策略包括水平扩展阶段和垂直扩展阶段。Chen [13]等人借鉴 IPO 策略提出了类 IPO 策略，该策略只考虑了两两组合的情况，本文将在类 IPO 策略的基础上提出改进的类 IPO 策略，并将其与改进后的 CS-SPSO 算法相结合来生成可覆盖任意维度的组合测试用例集。图 2 即为改进的类 IPO 策略的整体框架。

改进的类 IPO 策略利用等价类分析和约束条件对因素进行约束处理，以选取代表取值进行组合，有效减少测试组合数目；将用例的错误检出率 C_i 和响应时间 T_i 作为评估指标，以减少排序的随机性，提高策略运行效率，其中具体的评估公式如下：

$$\rho = 0.6C_i + 0.4T_i \tag{10}$$

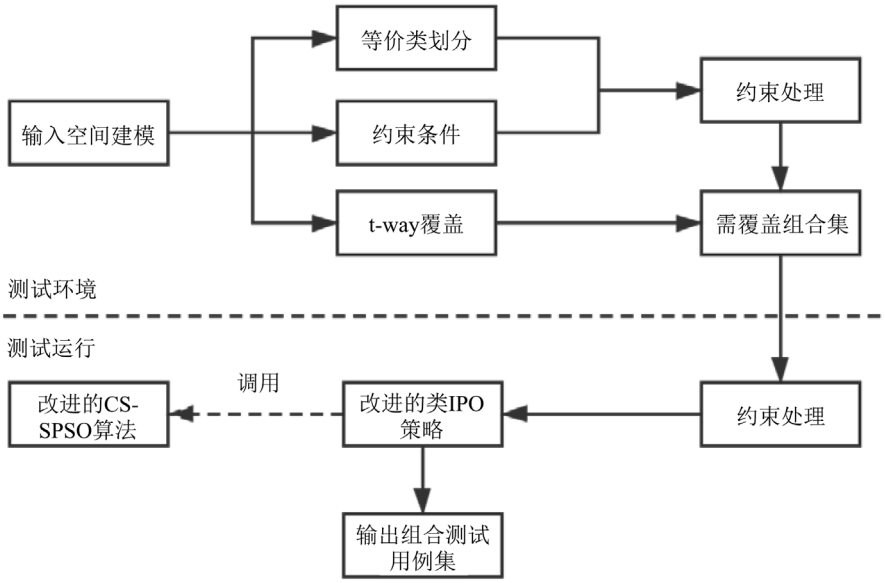


Figure 2. Improved IPO strategy framework
图 2. 改进的类 IPO 策略框架

改进的类 IPO 策略的具体算法伪代码如下所示：

算法 1： 改进的类 IPO 策略
输入：需覆盖组合集 UncovCombset，组合测试覆盖强度 t-way，因素个数 n，各因素取值 N_i
输出：测试用例 T

1. T = null;
2. 根据因素的评估指标进行非递增排序;
3. 将前 t 个因素取值进行组合，得到 UnCombset;
4. while(UnCombset≠null | UncovCombset≠null) {
5. 随机选择 UnCombset 中的任意一个组合 combset;
6. 将 combset 作为改进的 CS-SPSO 算法的输入值生成测试用例 t;
7. 计算测试用例 t 覆盖的组合 covcombset$_t$;
8. UncovCombset=UncovCombset-covcombset$_t$;
9. UnCombset=UnCombset-combset;
10. $T = T \cup \{t\}$; }
11. while(UncovCombset≠null) {
12. 随机选择 UncovCombset 中的任意一个组合 combset;
13. 重复 6~8 的操作;
14. $T = T \cup \{t\}$; }
15. return T;

改进的类 IPO 策略不仅提升了其运行效率还考虑了可变力度组合测试，可以满足任意强度的覆盖表。

4. 实验分析

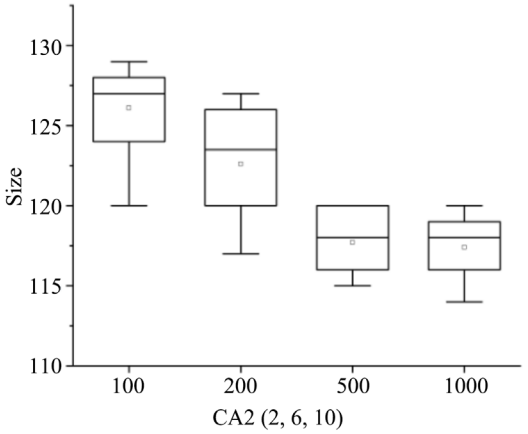
为了验证本文提出的基于改进粒子群算法的布谷鸟搜索优化算法(CS-SPSO)的有效性，将在 mac 操作系统的 Idea 工具上，采用 Java (JDK 1.8)语言编程分别实现 CS-SPSO 算法与基本的 PSO 算法和 CS 算法的对比，以及基本 IPO 策略和改进的类 IPO 策略的对比，以此来验证改进后算法的性能。本文采用 10 个具有代表性、复杂程度不同且组合维度不同的实例(见表 1)进行实验分析，其中有覆盖矩阵(CA)和混合覆盖矩阵(MCA)各 5 组。

Table 1. 10 coverage tables used in the experiment
表 1. 实验采用的 10 个覆盖表

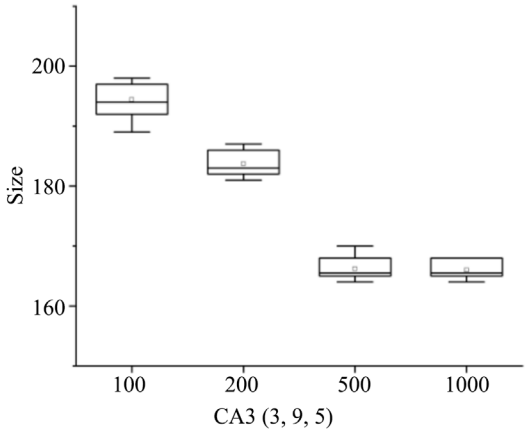
CA	MCA
CA1(2,4,3)	MCA6$(2,4,8^2 3^2)$
CA2(2,6,10)	MCA7$(2,8,4^3 3^2 2^3)$
CA3(3,9,5)	MCA8$(3,7,5^1 3^4 2^2)$
CA4(3,5,7)	MCA9$(4,7,5^2 4^3 3^2)$
CA5(4,8,5)	MCA10$(4,6,8^2 7^2 6^2)$

4.1. 参数设置

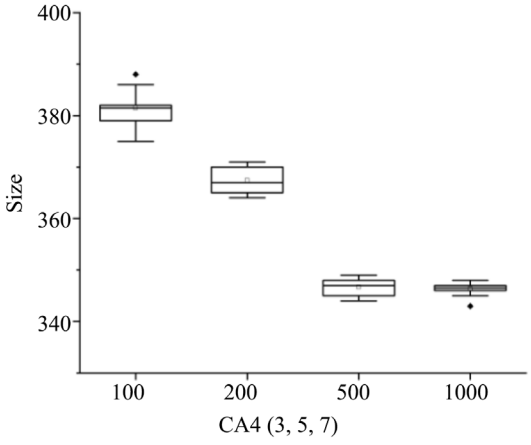
用 CS-SPSO 算法生成 t-way 组合测试用例集时，既要考虑基本粒子群算法的参数，又要考虑布谷鸟搜索算法的参数。本文算法的设置将参考文献[21] [22]：学习因 $c_1 = c_2 = 2$， r_1 和 r_2 是[0, 1]内的随机数，子种群个数 $N = 5$， $w_{max?} = 0.9$， $w_{min?} = 0.4$， $Pa = 0.25$。其中种群的迭代次数 T_{max} 的选取对算法有一定的影响。当 T_{max} 过大时，会消耗大量时间而实际测试用例规模不会减少太多。当 T_{max} 过小时，无法达到最小测试用例集的要求。所以将选择 CA2，CA3，CA4，MCA7，MCA8，MCA9 进行实验来找到实际的最优迭代次数。



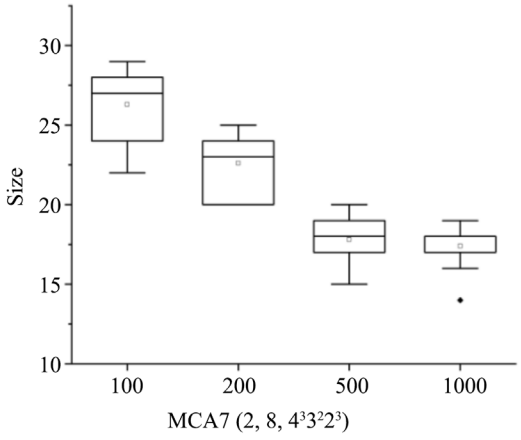
(a) CA2 的测试用例集规模分布图



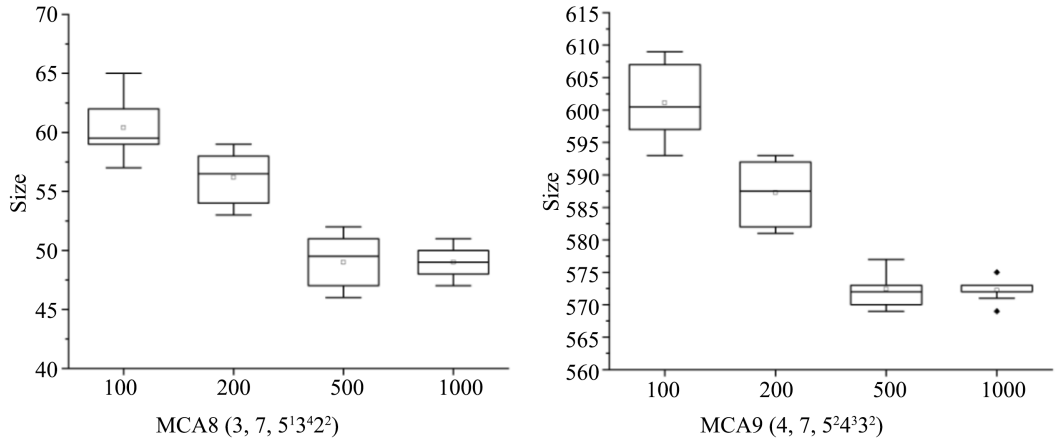
(b) CA3 的测试用例集规模分布图



(c) CA4 的测试用例集规模分布图



(d) MCA7 的测试用例集规模分布图

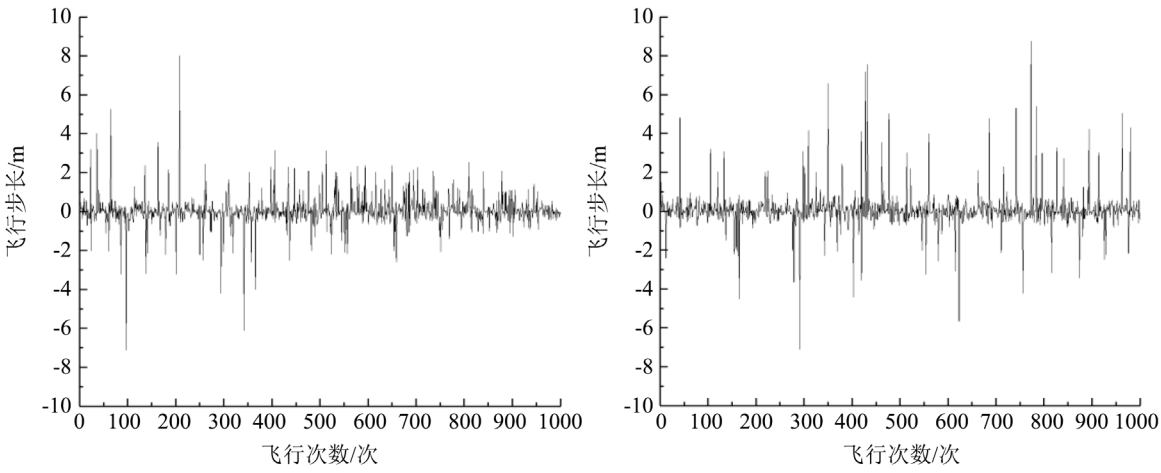


(e) MCA8 的测试用例集规模分布图　(f) MCA9 的测试用例集规模分布图

Figure 3. Size distribution of the combined test case set under different iterations
图 3. 不同迭代次数下组合测试用例集规模分布

经过图 3 的对比分析发现，增加迭代次数可显著降低测试用例集规模，但是当达到一定程度时，测试用例集规模会趋于稳定。对比 T_{max} 取 500 和 1000 的数据可知，两者平均测试用例集数很接近，因此考虑时间因素取 500 为宜。

步长 ε 是布谷鸟搜索算法中的重要参数，可影响算法的收敛速度。基本布谷鸟搜索中的步长是固定值，当步长过大时，算法后期收敛速度慢；当步长过短时，算法会陷入局部最优，因此本文提出了根据迭代次数进行动态调节的步长 ε，下面图 4 是 *Lévy* 飞行步长的对比图。



(a) 改进 CS 算法 1000 次 Lévy 飞行步长图　(b) CS 算法 1000 次 Lévy 飞行步长图

Figure 4. Comparison of improved CS and CS algorithm *Lévy* flight step size
图 4. 改进 CS 算法和 CS 算法 *Lévy* 飞行步长对比图

经过图 4 的对比可证明其动态步长的有效性，在算法后期可加快其收敛速度，提高算法的搜索性能。

4.2. PSO、CS 和改进的 CS-SPSO 算法的比较

为了规避 CS-SPSO 算法执行过程中随机因素对实验结果的影响，故对每组实例独立运行 100 次取平均值作为实验的对比数据。

表 2 是从测试用例集规模和算法运行时间这两个方面对不同算法进行比较。从用例规模上看，除了覆盖表 CA1、MCA9 以外，基于 CS-SPSO 算法在总体用例规模上都优于基本的 PSO 算法和 CS 算法。在因素的取值个数较多或 t-way 维度比较高的覆盖表中，其优势更加明显，例如 CA5、MCA10 等，经对比可发现本文提出的 CS-SPSO 算法对缩减测试用例集规模有一定的效果。

Table 2. Comparison of PSO, CS and improved CS-SPSO algorithms
表 2. PSO、CS 和改进的 CS-SPSO 算法的比较

Covering Array	PSO		CS		CS-SPSO	
	size	time/s	size	time/s	size	time/s
CA1	9	10.8	9	11.2	9	9.7
CA2	128.5	184.1	127.4	186	117.6	172.3
CA3	178	281.4	179.7	283.9	166.2	278.3
CA4	351.9	317.3	350.4	309.3	346.7	264.9
CA5	940.1	1843.7	937.5	1857.3	819.9	1427.5
MCA6	67.3	85.4	68.5	86.9	64	79.4
MCA7	24.3	267.5	22.9	268	17.8	250
MCA8	52.7	49.1	56.1	53.4	49	46.4
MCA9	393.5	867.2	412.3	892.6	572.4	796.4
MCA10	5521.8	17951.6	5597.6	17821	4936	13829.9

从时间性能上看，不考虑 CA1、MCA9 这些在用例规模上不占优势的覆盖表，CS-SPSO 相较于 PSO 算法和 CS 算法具有一定优势，在 CA5、MCA10 这些维度比较高的覆盖表上其优势更加明显。由此可见，本文提出的 CS-SPSO 算法可有效缩减执行时间。

4.3. 基本 IPO 策略和改进的类 IPO 策略的对比

在执行过程中，为了避免算法因随机因素对结果产生的影响，因此，本实验将对各实例覆盖表独立运行 100 次，将求取的平均值作为参考数据。

本文将利用基本 IPO 策略和改进的类 IPO 策略与改进的 CS-SPSO 算法相结合生成测试用例的方法来验证改进后的类 IPO 策略的有效性。

Table 3. Comparison of basic IPO strategy and improved IPO strategy
表 3. 基本 IPO 策略和改进的类 IPO 策略的比较

Covering Array	基本 IPO 策略		改进的类 IPO 策略	
	size	time/s	size	time/s
CA1	9	9.3	9	9.7
CA2	128.5	165.1	117.6	172.3
CA3	178	272	166.2	278.3
CA4	351.9	289.3	346.7	264.9
CA5	940.1	1852.7	819.9	1427.5
MCA6	67.3	75.4	64	79.4
MCA7	24.3	247.4	17.8	250
MCA8	49	45.9	49	46.4
MCA9	472.3	859.2	572.4	796.4
MCA10	5521.8	14921.4	4936	13829.9

表 3 也是从测试用例集规模和算法运行时间这两个方面对不同算法进行比较。从用例规模上看，除了覆盖表 CA1、MCA8、MCA9 以外，改进的类 IPO 策略在总体用例规模上都优于基本的 IPO 策略。在复杂度较高的覆盖表中，优势更加明显，例如 CA5、MCA10 等。从时间性能上看，在复杂度较高的 CA5 和 MCA10 覆盖表中，本文提出的改进的类 IPO 策略可有效缩减执行时间。

综上所述，本文提出的基于改进粒子群算法的布谷鸟搜索优化算法和改进的类 IPO 策略，在因素的取值个数较多或 t-way 维度比较高的情况下，其生成测试用例集规模和算法执行时间上都具有一定的优势。

5. 结束语

为了解决组合测试的 NP 问题、因素选取的随机性问题以及算法种群多样性差、执行时间过长等问题，本文提出一种基于改进粒子群算法的布谷鸟搜索优化算法(CS-SPSO)的组合测试用例集生成方法。种群的划分和子种群间粒子的交流增强了种群的多样性，还有效避免了 CS 算法利用随机函数易导致初始鸟巢的离散程度低、位置分配不够均匀的问题，布谷鸟搜索中步长的动态调节解决种群寻优后期震荡，收敛速度慢的问题。其中，改进的类 IPO 策略利用约束处理和评估指标减少排序的随机性，提高策略运行效率还能使其可应用于任意强度的覆盖表。但本文提出的方法在小种群的运算上还存在一些局限，在未来的工作中，可对初始种群划分、子种群并行化计算等方面进行深入研究。

基金项目

国家自然科学基金资助项目(No. 61502430)；浙江省重点研发计划项目(No. 2020C03094)；浙江省自然科学基金青年基金(No. LQ20F050010)。

参考文献

[1] 申情, 蒋云良, 沈张果, 等. 基于组合混沌遗传算法的最小测试用例集生成[J]. 电信科学, 2016, 32(6): 93-102.

[2] 宋晓秋, 梁凡. 两两组合测试用例生成的遍历搜索算法[J]. 计算机工程与设计, 2019, 40(2): 433-437.

[3] Lei, Y., Kacker, R., Kuhn, D.R., *et al*. (2008) IPOG/IPOG-D: Efficient Test Generation for Multi-Way Combinatorial Testing. *Software Testing, Verification and Reliability*, **18**, 24. https://doi.org/10.1002/stvr.381

[4] 艾华, 宋晓秋, 安恒. 简单循环约减三三组合测试用例生成方法[J]. 计算机工程与设计, 2018, 39(12): 3728-3733.

[5] Nie, C.H. and Leung, H. (2011) A Survey of Combinatorial Testing. *ACM Computing Surveys*, **43**, 1-29. https://doi.org/10.1145/1883612.1883618

[6] Sabharwal, S., Bansal, P., Mittal, N., *et al*. (2016) Construction of Mixed Covering Arrays for Pair-Wise Testing Using Probabilistic Approach in Genetic Algorithm. *Arabian Journal for Science and Engineering*, **41**, 2821-2835. https://doi.org/10.1007/s13369-015-2022-1

[7] 戚荣志, 王志坚, 黄宜华, 等. 基于 Spark 的并行化组合测试用例集生成方法[J]. 计算机学报, 2018, 41(6): 1064-1079.

[8] 曾梦凡, 陈思洋, 张文茜, 等. 利用蚁群算法生成覆盖表: 探索与挖掘[J]. 软件学报, 2016, 27(4): 855-878.

[9] 王小银, 王曙燕, 孙家泽. 基于蚁群算法的三三组合测试用例集的生成[J]. 计算机应用研究, 2015, 32(11): 3328-3331.

[10] Chandra, P., Subhash, T., Vrushali, K., *et al.* (2018) A Critical Review on Automated Test Case Generation for Conducting Combinatorial Testing Using Particle Swarm Optimization. *International Journal of Engineering & Technology*, **38**, 22-28. https://doi.org/10.14419/ijet.v7i3.8.15212

[11] 包晓安, 鲍超, 金瑜婷, 等. 基于动态调整简化粒子群优化的组合测试用例生成方法[J]. 计算机科学, 2018, 45(11): 199-203.

[12] Kennedy, J. (2011) Particle Swarm Optimization. *Proceedings of* 1995 *IEEE International Conference of Neural Networks*, **4**, 1942-1948.

[13] 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.

[14] 包晓安, 杨亚娟, 张娜, 等. 基于自适应粒子群优化的组合测试用例生成方法[J]. 计算机科学, 2017, 44(6): 177-181.

[15] Mahmoud, T. and Ahmed, B.S. (2015) An Efficient Strategy for Covering Array Construction with Fuzzy Logic-Based Adaptive Swarm Optimization for Software Testing Use. *Expert Systems with Applications*, **42**, 8753-8765. https://doi.org/10.1016/j.eswa.2015.07.029

[16] 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.

[17] Yang, X.S. and Deb, S. (2010) Cuckoo Search via Levy Flights. *Mathematics*, **22**, 210-214.

[18] Yang, X.S. and Deb, S. (2013) Multiobjective Cuckoo Search for Design Optimization. *Computers & Operations Research*, **40**, 1616-1624. https://doi.org/10.1016/j.cor.2011.09.026

[19] Yang, X.S. and Deb, S. (2010) Engineering Optimization by Cuckoo Search. *International Journal of Mathematical Modelling and Numerical Optimisation*, **1**, 330-343. https://doi.org/10.1504/IJMMNO.2010.035430

[20] Tai, K.C. and Lei, Y. (2002) A Test Generation Strategy for Pairwise Testing. *IEEE Transactions on Software Engineering*, **28**, 109-111. https://doi.org/10.1109/32.979992

[21] 张煜培, 赵知劲, 郑仕链. 利用多目标 PSO 优化的累积时延和信道容量联合优化的频谱切换算法[J]. 电信科学, 2018, 34(7): 62-71.

[22] Ahmed, B.S., Abdulsamad, T.S. and Potrus, M.Y. (2015) Achievement of Minimized Combinatorial Test Suite for Configuration-Aware Software Functional Testing Using the Cuckoo Search algorithm. *Information and Software Technology*, **66**, 13-29. https://doi.org/10.1016/j.infsof.2015.05.005