软件确保智能测试用例生成PSO算法进展研究

耿 技[1]，聂 鹏[1,2]，秦志光[1]

(1. 电子科技大学计算机科学与工程学院 成都 611731；2. 江西财经大学现代教育技术中心 南昌 330013)

【摘要】测试用例生成是软件测试的重要环节，是软件确保的重要组成部分，其中启发性算法是近年来测试用例自动生成研究领域的热点。对启发性测试用例生成的新方法PSO进行了介绍和分析，详细讨论了PSO算法适应度函数、PSO算法早熟与局部最优、种群规模对PSO算法的影响以及PSO参数优化问题，并将PSO与GA算法进行了对比分析。展望了PSO测试用例生成算法的未来研究方向，指出PSO测试用例生成算法目前应重点解决测试用例规模优化、早熟抑制和参数优选等问题。

关 键 词 启发性算法； 粒子群优化； 软件确保； 软件测试； 测试用例生成

中图分类号 TP311 文献标识码 A doi:10.3969/j.issn.1001-0548.2012.06.017

Overview of PSO for Automatic Test Case Generation in Software Assurance

GENG Ji[1], NIE Peng[1,2], and QIN Zhi-guang[1]

(1. School of Computer Science & Engineering, University of Electronic Science and Technology of China Chengdu 611731;

2. Center of Modern Education Technology, Jiangxi University of Finance and Economics Nanchang 330013)

Abstract The automatic test case generation is a key phase of software testing and an important part of software assurance. The study on the heuristic algorithms is an emerging area of the automatic test case generation in recent years. The new heuristic algorithm of PSO for the test case generation is reviewed and analyzed. Key issues on the PSO test case generation are discussed, including PSO fitness functions, PSO premature convergence and local optimum, swarm size impact, and parameter optimization. A contrastive analysis of PSO and GA in software testing is presented in detail. Finally, the future development of PSO test case generation algorithms is prospected including the test case swarm size optimization, the premature restraining, and the parameter optimization.

Key words heuristic algorithm; particle swarm optimization; software assurance; software testing; test case generation

文献[1]首次提出了“软件确保”的概念，指出软件确保是旨在提高软件质量的任何实践、技术和工具，因此一切有利于提高软件质量的方法都可以称之为软件确保方法。随着软件确保研究的发展，软件确保概念已不再局限于“软件质量”，而更进一步扩展到S^3R模型[2]。其中，软件测试是软件确保的有效手段之一，目的是尽可能发现并改正被测试软件中的错误，提高软件的可靠性，对软件可靠性保证具有极其重要的意义。

IEEE计算机协会对软件测试的定义为通过人工测试或自动测试手段对软件质量进行度量，其目的是检验被测软件实际运行结果是否与软件设计初衷相一致。启发性软件测试用例生成算法在近些年中被广泛研究，包括遗传算法(genetic algorithm，GA)[3-6]、神经网络(neural network，NN)[7]、蚁群算法(ant colony algorithm，ACA)[8-9]、Petri网(petri net，PN)[10]。其中，GA算法是结构测试数据自动生成研究工作中的代表性算法。

在启发性算法中，粒子群算法(particle swarm optimization，PSO)[11]测试用例生成算法具备算法简单、鲁棒性好的特点，具有很好的导向性和收敛性，且使用成本较低。

本文首先对PSO测试用例生成算法思想进行研究，其次对算法中关键性问题进行分类比较，然后对PSO与GA进行比较，最后进行归纳和展望。

收稿日期：2010－12－22； 修回日期：2011－08－30

基金项目：国家自然科学基金(60973118, 61133016)；国家863计划(2011AA010706)

作者简介：耿技(1963－)，男，教授，主要从事系统软件、软件确保、软件智能测试方面的研究.

1 粒子群算法(PSO)

1.1 算法原理

PSO是一种启发式计算技术，最早由Kennedy和Eherhart通过对动物社会行为的观察而提出，是一种新的群体智能优化算法。

PSO测试用例生成算法将测试对象参数约束集作为算法的解空间；覆盖测试目标的潜在测试用例集合，称之为“粒子群”；每个测试用例称之为“粒子”，并以一定速度在D维解空间中搜索测试目标。粒子对环境的适应度(fitness value)由量化评估函数进行评估。在算法迭代过程中，粒子通过跟踪两个“极值”来更新自己。第一个极值为粒子本身所找到的最优解，称之为个体最优(p_{best})；第二个极值是整个粒子群找到的最优解，为全局最优(g_{best})。

PSO标准算法为：

在D维搜索空间中，存在 m 个粒子，其中第 i 个粒子的位置为 $x_i=(x_{i1},x_{i2},\cdots,x_{id})$， $i=1,2,\cdots,m$，其速度为 $v_i=(v_{i1},v_{i2},\cdots,v_{id})$； $p_i=(p_{i1},p_{i2},\cdots,p_{id})$ 为粒子 i 在D维空间的个体最优位置， $p_g=(p_{g1},p_{g2},\cdots,p_{gd})$ 为粒子群整体在D维空间的全局最优位置。

粒子速度与位置更新方程分别为：

$$v_{id}=\begin{cases} wv_{id}+c_1r_1(p_{id}-x_{id})+c_2r_2(p_{gd}-x_{id}) & v_{id}\leqslant v_{\max d} \\ v_{\max d} & v_{id}>v_{\max d} \end{cases} \tag{1}$$

$$x_{id}=x_{id}+v_{id} \tag{2}$$

PSO算法中各个参数及其含义如表1所示。

表1 PSO算法参数

参 数	含 义
w	惯性权重(inertia weight)
c_1、c_2	加速常数(acceleration constants)
r_1、r_2	[0,1]范围内的随机函数
x_{id}	第i个粒子在第d维空间的位置
v_{id}	第i个粒子在第d维空间的速度
p_{id}	第i个粒子在第d维空间经历的最好位置
p_{gd}	粒子群经历的最好位置在第d维空间上的分量
$v_{\max d}$	粒子在第d维空间的最大速度
$G_{\max}$	算法允许的最大迭代次数

式(1)中，v_{id} 为粒子先前的速度；$c_1r_1(p_{id}-x_{id})$ 为个体“认知”部分，表示单个粒子本身的思考，$c_2r_2(p_{gd}-x_{id})$ 为“社会”部分，表示粒子间的信息共享与合作。式(2)表示单个粒子对自身位置的更新，其中等号左边部分为粒子的新位置，右边部分分别为粒子历史位置和运动速度。

PSO中的“认知”部分可由“影响法则”[12]解释，即一个得到加强的随机行为在将来可能出现的概率更高，同时获得正确知识的能力也会得到加强。

“社会”部分可由代理[13]加强概念解释，当观察者观察到一个模型在加强某一行为时，将增加模型对该行为的实施概率，即粒子本身的认知将被其他粒子所模仿。

1.2 标准PSO算法流程

标准PSO算法流程如下：

1) 随机初始化粒子群的位置和速度；

2) 计算粒子的适应值；

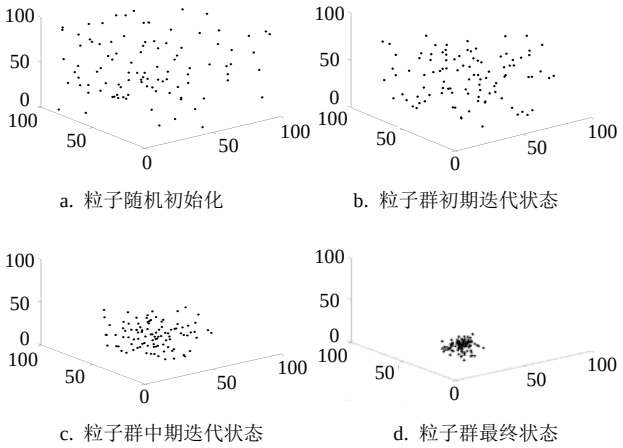
3) 将粒子的适应值与其历史最好位置p_i的适应值进行比较，若较好，则将其作为当前的最好位置；

4) 将粒子适应值与全局历史最好位置p_g的适应值进行比较，若较好，则将其作为当前的全局最好位置；

5) 如未达到结束条件(通常为足够好的适应值或达到一个预设的最大代数)则执行步骤6)，否则算法结束；

6) 根据式(1)和式(2)对粒子的速度和位置进行更新，产生新粒子群并返回步骤2)。

图1为粒子群在三角分类单路径覆盖试验中不同阶段的粒子分布图。



a. 粒子随机初始化　b. 粒子群初期迭代状态

c. 粒子群中期迭代状态　d. 粒子群最终状态

图1 三角分类单路径覆盖PSO算法粒子分布图

1.3 PSO群体智能

PSO是群体智能的具体实例，天然具备与环境交互并从中得到全局最优解的能力，具有如下特点：

1) 分布式控制，即无中心控制，因此能够更好地适应软件测试环境，表现出较强的鲁棒性，不会因某些测试用例异常而影响整个问题的求解；

2) 群体智能的改变是测试用例以非直接通信的方式在整个种群进行信息传输与合作结果，通信开销对个体数目增加不敏感，具有较好的可扩充性；

3) 群体中每个测试用例的能力或遵循的行为

规则非常简单，群体智能实现比较方便；

4) 群体行为的复杂性是源于简单个体在交互过程产生的突发智能，因此群体具有自组织性。

1.4 适应度函数

适应度函数用于评估测试用例在搜索空间中对测试目标的接近程度，是PSO自动测试用例算法中的重要组成部分。适应度函数的定义可以分为两类。

1) 面向分支的适应度函数(BOFF)。

面向分支适应度函数在分支节点前插装，需对插装节点进行语义分析，但无需插装该节点后的分支。

控制流图是对程序进行描述的抽象数据结构，是由节点和边构成的有向图，其中节点代表代码块，边代表控制流在节点间的转移。通常，控制流图包括一个入口节点和一个出口节点，并通过它们控制程序的输入和输出。令有向图$C=(N,A,s,e)$为待测程序Q的控制流图，其中N为有向图C的节点集，A为N上的关系表达，s和e分别为Q的入口点和出口点。有向图C中存在路径$P_s=\{n_1,n_2,\cdots,n_q\}$，其中$n_1=s$，$(n_i,n_{i+1})\in A$，$1\leqslant i<q$，$q$为路径$P_s$中的节点数。此时测试用例(TC)对$P_s$的适应度由分支预测度(BP)给出，形式为$E_1$ op E_2，E_1与E_2为数学表达式，$op\in\{<,\leqslant,>,\geqslant,=,\neq,\&\&,\|,!\}$；$E_1$ op E_2可进一步表达为Fre 10，F为分支函数，其运算关系详见文献[14]。

BOFF适应度算法步骤为：

① 对程序 Q 进行语法分析，构建抽象语法树AST；并通过 AST 产生控制流图 C；

② 从 C 中抽取分支节点 n_i，构成分支节点路径集 $P=\{P_1,P_2,\cdots,P_m\}$，其中 m 为集合 P 中不同的路径数，$P_s=\{n_1,n_2,\cdots,n_q\}$，$n_i\in N$；

③ 对节点 n_i 进行解析，生成 P_j 对应的分支函数序列集合 $F=\{F_1,F_2,\cdots,F_m\}$，其中 m 为集合 F 中不同分支函数序列数，每一个 $F_s=\{f_1,f_2,\cdots,f_q\}$ 唯一对应一条路径 P_s，f_i 为 P_s 路径上节点 i 对应的分支函数值；

④ 计算路径 F_s 分支函数叠加适应度 fitness：

$$\text{fitness}_i=\begin{cases}f_i & \text{节点取“真”值时}\\ \theta_i-f_i & \text{节点取“假”值时}\end{cases}\tag{3}$$

$$\text{fitness}=\sum_{i=1}^{m}\text{fitness}_m\tag{4}$$

其中 θ_i 为路径 F_s 中 f_i 的阈值上限。

2) 面向路径的适应度函数(POFF)。

面向路径适应度函数在分支节点后插装，需对插装节点分支结构进行解析，但不依赖于插装节点本身的语义分析。

根据控制流图分支节点，测试用例适应度值由用例与目标间的距离给出。令 $\text{target}=\{P_0,P_1,\cdots,P_{n-1}\}$ 为目标路径序列，其中 $P_i(0\leqslant i\leqslant n-1)$ 为路径上的条件语句或分支节点；令 $\text{track}=\{t_0,t_1,\cdots,t_{m-1}\}$ 为测试用例执行路径，其中 $t_i(0\leqslant i\leqslant m-1)$ 为执行路径覆盖的条件语句或分支节点涉及的分支路径。测试用例与目标间距定义为：

$$\text{Distance(track,target)}=\text{Length(target)}-\text{Similarity(track,target)}\tag{5}$$

式中，Length(•) 返回目标路径中节点的个数用 n 表示；Similarity(track,target) 为测试用例执行路径和目标路径间的近似度，值域为 $[0,n]$，值越高表示两者相似度越高；Distance(track,target) 值域为 $[0,n]$，其值越高表示测试用例对于目标路径的覆盖度越低。

POFF适应度算法步骤为：

① 对程序 Q 进行语法分析，形成抽象语法树AST；

② 通过 AST 产生控制流图 C；

③ 从 C 中抽取分支节点 P_i，构成分支节点路径集 $P=\{\text{target}_1,\text{target}_2,\cdots,\text{target}_m\}$，其中 m 为集合 P 中不同的目标路径数，$\text{target}_s=\{p_1,p_2,\cdots,p_{n-1}\}$，$p_i(0\leqslant i\leqslant n-1)$；

④ 在每个节点涉及的分支 t_i 下插装测试用例执行路径跟踪函数，最终形成测试用例执行路径 $\text{track}_i=\{t_0,t_1,\ldots,t_{m-1}\}$；

⑤ 计算 $\text{fitness}_i^s=\text{Distance}(\text{track}_i,\text{target}_s)$，$\text{fitness}_i^s$ 为第 i 个测试用例对第 s 条目标路径的适应度。

BOFF在解空间搜索最优解时比POFF更加细致，有利于用例更细致地搜索目标；但当变更测试目标时需要更新适应度函数结构，会增加语法分析和计算成本。POFF在搜索最优解时比BOFF更加强调实际覆盖路径和目标路径间的差异性，当变更测试目标时无需更新适应度函数结构，不会增加额外的语法分析和计算成本。

2 PSO测试用例算法关键性问题

2.1 早熟与局部最优

PSO以其计算迅速和易用性优于传统算法，但是PSO如同其他智能算法一样存在早熟，存在过早产生局部最优解的现象，特别在大规模优化问题中表现比较突出。

文献[15-18]在PSO中引入变异技术，用于抑制算法过快收敛到局部最优解。研究表明，PSO算法的早熟收敛与粒子群多样性有着密切的关系。好的PSO算法应能使多样性曲线保持比较大的震动频率和振幅，尤其是算法前期的震动频率和振幅，这将有利于算法寻找全局最优和局部最优的平衡点。

文献[19]将模糊聚类引入PSO，使用C-Means (FCM)聚类方法对粒子群进行分类，禁止近亲繁殖，提高粒子群多样性，较标准PSO迭代次数减少67%。

文献[20]通过量化粒子条件相似度的方式评估粒子多样化水平，并替换多样化水平低于阈值的粒子，进而增强粒子群多样性，避免早熟。

2.2 粒子群规模

PSO是一种对复杂优化问题解空间进行探索的启发式搜索方法，依赖于粒子群通过迭代操作共享信息。随着问题规模的扩大，迭代次数也会随之变大；当系统测试算法使用较为复杂的适应度函数时，这一现象将更为恶化。

针对该问题，有3种可选改进：1) 使用轻量级的适应度函数，从而减少单个粒子更新的时间消耗；2) 动态改变参与迭代的粒子群规模，仅使得必要的粒子参与实际运算；3) 并行PSO算法。

方法1在无更优可替代适应度函数时表现出很大的刚性；并且随测试目标不同，适应度函数的可约简性也不同，具体实现复杂，鲁棒性差。

方法2具有较好的普适性，操作简单，鲁棒性好，分为3个亚类：1) 粒子群规模递减型[21-22]。该类方法认为PSO算法本身具有良好的收敛性，当粒子群规模固定时，这种收敛性将使粒子群中粒子相互覆盖而产生冗余，因此在无损粒子信息的前提下，剔除重复的粒子将显著降低PSO的计算成本。2) 粒子群规模递增型[21]。该类方法认为PSO早期迭代中，应使用较少的粒子，随着迭代进行，这些粒子会很快向目标区靠近，从而自动地增加了粒子群密度；在第二阶段，当粒子群接近最优解时，已经具有较高密度，因此仍可覆盖全局最优解。3) 粒子群规模自适应型。该类方法认为粒子群规模应该在收敛性和粒子多样性之间进行动态平衡：单纯递减型将导致算法收敛过快，而陷入局部最优解；单纯递增型会在算法后期产生冗余粒子，浪费计算资源。文献[23-24]将粒子群分割为不同的簇，并对簇进行密度评估从而跟踪动态调整群规模，该方法较其他多目标PSO(MOPSO)算法迭代数平均减少70%，时间消耗平均减少87%。

方法3将粒子群分为多个可并发执行的簇[25-28]，对每个簇设定相关参数，使每个进程独立地对簇进行单独进化，从而降低算法的整体时间成本。

2.3 PSO参数优化

2.3.1 惯性权重优化

惯性权重w其值既可以为常数，也可以为变量，用于控制粒子对搜索空间的探索能力。文献[21]在惯性因子w对优化性能的影响研究中认为，较大的w值使粒子运动速度较快，并可快速找到全局最优邻接粒子，利于跳出局部最优；较小的w值则有利于粒子较细致地对局部空间进行搜索，有利于算法的收敛。采用惯性权重自适应策略抑制PSO算法早熟，自适应算法通过对w线性减小，动态调整该参数。该调整使算法在迭代初期探索能力增强，可以连续搜索新区域；迭代后期开发能力增强，使粒子群在可能解的周围进行细致搜索。

2.3.2 加速权重优化

式(2)中的c_1、c_2分别用于控制粒子获取其个体最好位置和全局最好位置；选取较小的参数值将限制粒子群移动活性，而放大参数值则将导致粒子分布过于发散。文献[29]发现增加c_1和c_2的值将导致粒子在最优解附近产生较高频率的振荡，不利于算法收敛，并建议$c_1+c_2\leqslant 4$。

文献[22]用实验表明c_1和c_2的值为常数时可以获得较好的解，但不一定为2；群体的初始化也对算法性能有所影响，但不十分明显。文献[30]提出收缩因子的概念，该方法描述了一种选择$w(t)$、c_1、c_2值的方法，以确保算法收敛。

2.3.3 速度优化

在每次迭代中，PSO算法将动态调整粒子在搜索空间各维的速度分量。速度分量随机产生，目的在于向粒子运动引入随机扰动，使其能在解空间进行更大范围的搜索。

文献[31]表明，速度上限的动态调整有助于改善PSO的性能。文献[32]推荐使用式(5)对最大速度进行动态调整;其中N为第d维空间分割数，$x_{\max D}$、$x_{\min D}$为第d维空间中粒子位置的最大值和最小值。

$$v_{\max D}=(x_{\max D}-x_{\min D})/N \tag{5}$$

3 PSO与GA比较

文献[33]系统论述了遗传算法(GA)，自此遗传算法被广泛应用于智能学习和优化问题。该算法是一种适应性搜索算法，是将自然界遗传基因的遗传过程与达尔文适者生存原则相结合，不断进化与淘汰可能解，最终得到最优解的过程。GA算法在测试

用例生成中的应用已经被广泛研究[3-6, 34-35]，这些方法存在算法复杂、参数不易设置问题；实验证明PSO较GA有更高的效率。

PSO和GA同属于适应性搜索算法，存在一些共同之处：1) 两者都需随机初始化种群；2) 都使用适应度值来评价系统；3) 都根据适应度值来进行搜索；4) 两个系统都不能保证一定找到最优解。

两者也具有明显差异：1) 标准PSO通过粒子个体位置更新产生新的粒子群，而无交叉(crossover)和变异(mutation)环节；2) PSO可通过控制个体粒子移动速度、种群密度、粒子间距等方式多样性地调节收敛过程，而GA仅通过种群多样性调节；3) PSO中粒子存在个体记忆特性p_{iD}和整体性记忆$p_{g\mathrm{D}}$，而GA算法无记忆。

就算法复杂度而言，由于GA算法中需要对TC进行排序，因此GA算法复杂度较高，为$O(T\log_2^{n^n})$至$O(Tn^2)$，其中n为种群大小，T为迭代次数。PSO算法无需对TC进行排序因此复杂度较低，为$O(Tnd)$，其中d为搜索空间的维数，通常在软件测试中$d \ll n$。

文献[36]对GA算法和PSO算法进行了比较，PSO与GA实验对比数据如表2所示。

表2　PSO与GA成等边三角形测试数据实验对比

粒子群大小	找到最优解迭代的次数					
	GA			PSO		
	最好	最差	平均	最好	最差	平均
100	3	443	161.5	3	13	8.4
120	4	507	204	5	11	8.2
140	13	282	116	5	8	6.8
160	14	220	110.5	1	10	6
180	2	122	48.7	2	10	5.9
200	2	142	55.2	2	8	5.4
220	18	116	66.2	1	8	4.8
240	3	156	48.2	1	7	4.1
260	9	120	62.6	2	7	4.4
280	11	260	89.5	2	9	5.6

实验证明，在三角形判定单路径覆盖测试中，PSO迭代次数在最好情况下大约是GA方法的1/3，最差情况下大约是GA方法的1/26。根据平均情况看，生成所需测试数据的迭代次数大约是GA的1/16，所耗费时间大约是GA的1/24；在三角形判定全路径覆盖测试中，PSO算法较单独生成测试数据所需时间大约是全路径生成所需要时间的2.48倍。

4　结 束 语

软件测试用例自动生成技术研究是软件确保过程的重要环节，自文献[1]提出“软件确保”起，软件测试技术便迅速成为软件确保不可分割的部分。本文就PSO测试用例生成算法进行了介绍，并重点分析了适应度函数、PSO早熟与局部最优、粒子群规模、PSO参数优化等4个关键问题，最后对PSO与GA算法进行了对比。

在近十几年中，包括遗传算法、蚁群算法、神经网络、Petri网等多种软件测试用例自动生成技术已经被广泛地研究讨论；粒子群算法(PSO)自动测试用例生成研究的相关工作在近几年成为新的关注点，但从测试用例算法优化角度进行的研究尚少。PSO测试用例生成算法可进一步加强在测试用例规模优化、早熟抑制和参数优选方面的研究工作。

参 考 文 献

[1] FUJII M S, A comparison of software assurance methods [C]//Proceedings of the 1st Annual Software Quality Assurance Workshop on Functional and Performance. [S.l.]: ACM Press, 1978:27-32.

[2] 方滨兴, 陆天波, 李超. 软件确保研究进展[J]. 通信学报, 2009, 30(2): 106-117.
FANG Bin-xin, LU Tian-bo, LI Chao. Survey of software assurance[J]. Journal on Communications, 2009, 30(2): 106-117.

[3] MCGRAW G, MICHAEL C, SCHATZ M. Generating software test data by evolution[J]. IEEE Transactions on Software Engineering, 2001(27): 1085-1110.

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

[5] GIRGIS M R. Automatic test data generation for data flow testing using a genetic algorithm[J]. Journal of Universal Computer Science, 2005, 11(5): 898-915.

[6] GHIDUK A S, HARROLD M J. Using genetic algorithms to aid test-data generation for data-flow coverage[C]//the 14th Asia-Pacific Software Engineering Conference. Nagoya, Japan: IEEE, 2007.

[7] ZHAO Rui-lian, LV Shan-shan. Neural-network based test cases generation using genetic algorithm[C]//the 13th Pacific Rim International Symposium on Dependable Computing. Melbourne, Victoria, Australia: IEEE, 2007: 97-100.

[8] 傅博. 基于蚁群算法的软件测试数据自动生成[J]. 计算机工程与应用, 2007, 43(12): 97-99.
FU Bo. Automated software test data generation based on ant colony algorithm[J]. Computer Engineering and Applications, 2007, 43(12): 97-99

[9] 陈明师, 刘晓洁, 李涛. 基于多态蚁群算法的测试用例自动生成[J]. 计算机应用研究, 2009, 26(6): 1347-1348.
CHEN Ming-shi, LIU Xiao-jie, LI Tao. Automated software test-data generation based on polymorphic ant colonies algorithm[J]. Application Research of Computers. 2009, 26(6): 1347-1348.

[10] ADJIR N, SAQUI-SANNES P D, RAHMOUNI M K. Time-optimal real-time test case generation using

prioritized time Petri nets[C]//the First International Conference on Advances in System Testing and Validation Lifecycle. Porto, Portugal: IEEE, 2009.

[11] KENNEDY J, EBERHART R. Particle swarm optimization[C]//Proceedings of IEEE International Conference on Neutral Networks. Perth, Australia: IEEE, 1995: 1942-1948.

[12] THORNDIKE E L. Animal intelligence: empirical studies[M]. New York: Mac Millan, 1911.

[13] BANDURA A. Social foundations of thought and action: a social cognitive theory[M]. New Jersey: Prentice Hall, 1986.

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

[15] PANT M, RADHA T, SINGH V P. A new diversity based particle swarm optimization using Gaussian mutation[C]// International Journal of Mathematical Modeling, Simulation and Applications. New Delhi, India: IEEE, 2008.

[16] PANT M, THANGARAJ R, ABRAHAM A. Particle swarm optimization using adaptive mutation[C]// Proceedings of the 19th International Conference on Database and Expert System Application. Turin, Italy: IEEE, 2008.

[17] THANGARAJ R, PANT M, ABRAHAM A. A new diversity guided particle swarm optimization with mutation [C]//World Congress on Nature and Biologically Inspired Computing. Coimbatore, India: IEEE, 2009.

[18] JIANG Shan-he, WANG Qi-shen, JIANG Ju-lang. Particle swarm optimization algorithm based on velocity differential mutation [C]//Chinese Control and Decision Conference. Guilin, China: IEEE, 2009.

[19] YANG Jun-jie, XUE Li-qin. Adaptive population differentiation PSO algorithm[C]//the Third International Symposium on Intelligent Information Technology Application. Nanchang, China: IEEE, 2009.

[20] XIE Xiao-feng, ZHANG Wen-jun, YANG Zhi-lian. Adaptive particle swarm optimization on individual level [C]//2002 6th International Conference on Signal Processing. Beijing, China: IEEE, 2002.

[21] SHI Yu-hui, EBERHART R C. Empirical study of particle swarm optimization[C]//Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway, New Jersey: IEEE, 1999.

[22] SUGANTHAN P N. Particle swarm optimiser with neighbourhood operator[C]//Proceedings of the Congress on Evolutionary Computation. Washington DC, USA: IEEE, 1999.

[23] LEONG W F, YEN G G. PSO-based multiobjective optimization with dynamic population size and adaptive local archives[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 2008, 38(5): 1270-1293.

[24] LEONG W F, YEN G G. Dynamic population size in PSO-based multiobjective optimization[C]//2006 IEEE Congress on Evolutionary Computation. Vancouver, BC, Canada: IEEE, 2006.

[25] KOH B, FREGLY B J, GEORGE A D, et al. Parallel asynchronous particles swarm for global biomechanical[J]. International Journal for Numerical Methods in Engineering, 2006, 67(4): 578-595.

[26] JIE Jing, Zeng Jian-chao, Han Chong-zhao. Self-organization particle swarm optimization based on information feedback[C]//Advances in Natural Computation, Second International Conference. Xi'an, China: Springer Press, 2006.

[27] MCNABB A W, MONSON C K, SEPPI K D. MRPSO: mapreduce particle swarm optimization[C]//Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. Atlanta, USA: ACM Press, 2007.

[28] YANG Dao-ping, ZHANG Kai, FAN Lin-bo, et al. A parallel chaos particle swarm optimization[C]//Proceedings of 2009 International Conference on Environmental Science and Information Application Technology. Wuhan, China: IEEE, 2009.

[29] OZCAN E, MOHAN C. Particle swarm optimization: surfing the waves[C]//Proceedings of Congress on Evolutionary Computation. Washington DC, USA: IEEE, 1999.

[30] CLERC M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization [C]//Proceedings of the Congress on Evolutionary Computation. Washington DC, USA: IEEE, 1999.

[31] FAN H, SHI Yu-hui. Study on Vmax of particle swarm optimization[C]//Proceedings of Workshop on Particle Swarm Optimization. Indianapolis, USA: Purdue School of Engineering and Technology, 2001.

[32] ABIDO A. Optimal power flow using particle swarm optimization[J]. International Journal of Electrical Power & Energy Systems, 2002, 24(7): 563-571.

[33] HOLLAND J. Adaptation in natural and artificial systems[M]. Ann Arbor, MI, USA: the University of Michigan Press, 1975.

[34] BOTTACI L. A genetic algorithm fitness function for mutation testing[C]//Proceedings of the SEMINALL-workshop at the 23rd International Conference on Software Engineering. Toronto, Canada: IEEE, 2001

[35] SHEN Xia-jiong, Wang Qian, Wang Pei-pei, et al. Automatic generation of test case based on GATS algorithm[C]//IEEE International Conference on Granular Computing. Nanchang, China: IEEE, 2009.

[36] LI Ai-guo, ZHANG Yan-li. Automatic generation method of test data for software structure based on PSO[J]. Computer Engineering, 2008, 34(6): 93-97.

编 辑 税 红