基于差分进化算法的组合测试用例集生成*

郭书杰[1a,2]，黄 明[1b]，梁 旭[1b]，焦 璇[3]

(1. 大连交通大学 a. 机械工程学院；b. 软件学院，辽宁 大连 116028；2. 91550 部队 指控中心，辽宁 大连 116023；3. 大连理工大学 管理学院，辽宁 大连 116024)

摘 要：为了减小所生成的组合测试用例集规模，提出了一种基于 one-test-at-a-time 策略的差分进化算法来求解该问题的方案。通过实验，比较了不同变异方式的优化性能，考察了各个算法参数对优化性能的影响，并与其他常用方法进行了对比。实验结果表明，与其他常用的组合测试用例生成方法相比，基于差分进化算法的生成方法能够生成规模更小的组合测试用例集。并且组合数越多，该算法的优势就越明显。

关键词：组合测试；测试用例生成；差分进化算法

中图分类号：TP311.56 **文献标志码**：A **文章编号**：1001-3695(2014)05-1449-03

doi:10.3969/j.issn.1001-3695.2014.05.039

Generate combination test cases with differential evolution algorithm

GUO Shu-jie[1a,2], HUANG Ming[1b], LIANG Xu[1b], JIAO Xuan[3]

(1. *a. Mechanical Engineering Institute, b. Software Technology Institute, Dalian Jiaotong University, Dalian Liaoning* 116028, *China*; 2. *Command & Control Center, PLA* 91550 *Troops, Dalian Liaoning* 116023, *China*; 3. *School of Management, Dalian University of Technology, Dalian Liaoning* 116024, *China*)

Abstract: For the same combined strength, a test suite with smaller scale is more superior. According to the characteristics of the problem, this paper proposed an approach based on one-test-at-a-time strategy differential evolution algorithm to solve the problem. By experimenting, it compared the optimum performance of different varying modes, observed and studied the effects of various parameters on the optimization algorithm performance. Comparative experiments with other commonly used methods show that the scale of combinatorial test suite generated by differential evolution algorithm is smaller.

Key words: combinatorial testing; test case generation; differential evolution algorithm

组合测试是一种基于规约的软件测试技术，可以在保证缺陷检测能力的基础上减少测试用例的规模。伴随着软件高度可配置化的发展趋势，组合测试已经成为一种应用广泛的有效测试手段。组合测试用例集的生成是一个 NP 完全问题，是组合测试的关键。近年来，组合测试用例集的生成问题引起了广大研究人员的关注，提出了不少解决方法[1]。王子元等人[2]通过对各因素间实际交互关系的关注，提出了一种新型的可变力度组合测试用例集生成方法；黄陇等人[3]提出了改进的 AETG 算法和 IPO 算法用于 *n*-way 组合覆盖测试用例生成；史亮等人[4]提出了一种解空间树模型，通过路径搜索来生成测试数据；Song 等人[5]基于交互树模型的组合测试集生成方法；陈翔等人[6]提出了一种基于粒子群优化的成对组合测试算法框架；查日军等人[7]提出了交叉熵方法和粒子群优化算法来生成两两组合测试数据；梁亚澜等人[8]分析了使用遗传算法生成覆盖表时，算法参数对结果的影响；McCaffrey[9]提出了一种使用遗传算法生成成对组合测试用例的方法。智能优化算法是解决 NP 完全问题的有效手段，越来越多的研究人员开始使用遗传算法、粒子群优化算法等智能优化方法来解决组合测试数据的生成问题。差分进化算法作为一种新兴的智能优化技术，在解决复杂优化问题时具有不俗的表现[10]，因此提出使用差分进化算法求解组合测试用例生成问题。

1 组合测试

软件系统可以看做一个处理器，它依据一定的规则对输入的数据进行处理，并输出处理结果。这些输入数据称为输入参数。统计结果显示，90% 以上的错误是由多个参数的交互作用触发的[11]。为了充分检测各个输入参数间的交互作用对系统的影响，提高错误检出率，提出了组合测试的概念。

定义 1 输入项。它是指影响软件系统处理流程或输出结果的因素，包括软件的配置参数、外部事件、输入数据等内容。

定义 2 可选值。它是一个输入项的可能取值。对于输入数据型输入项，通常采用边界值划分或等价类划分等黑盒测试技术对输入值进行处理，以便得到有限个离散的可选值。

定义 3 组合强度。给定一个测试用例集 *TR*，如果 *TR* 覆盖了任意 *t* 个输入项的全部组合形式，则称 *TR* 的组合强度为 *t*。

假设一个待测软件系统 *T* 有 *n* 个相互独立的输入项 $I = \{I_1, I_2, \cdots, I_n\}$，每个输入项包含有限个可选值，即 $V_i = \{v_{i1}, v_{i2}, \cdots, v_{in}\}$，则组合测试需要根据覆盖强度的要求生成组合测试用例集 *TR* 对系统进行测试，考虑到测试成本，*TR* 中测试用例的数量越少越好。

收稿日期：2013-06-24；**修回日期**：2013-08-03 **基金项目**：国家“863”计划资助项目(2012AA041402-4)

作者简介：郭书杰(1978-)，男，河南中牟人，工程师，博士研究生，主要研究方向为计算机集成制造和软件工程(shujieguo@126.com)；黄明(1961-)，男，吉林长春人，教授，博导，主要研究方向为计算机集成制造；梁旭(1974-)，女，天津人，教授，博士，主要研究方向为计算机管理信息系统；焦璇(1986-)，女，辽宁大连人，博士研究生，主要研究方向为信息管理与电子政务.

2 生成方案

2.1 差分进化算法

差分进化算法(differential evolution, DE)是一种简单有效的新型进化算法,已被成功应用于工程优化、机械设计、电力、环境、水利等多个领域[12~15]。标准差分进化算法包括初始化种群、变异操作、交叉操作、选择操作四个步骤。

1)初始化种群 随机产生 N 个个体构成初始种群。

2)变异操作 在每一代搜索中,DE 算法通过变异操作为当前种群中的每个个体 $x_i(g)$ 生成一个目标个体 $t_i(g)$(g 表示进化代数)。DE 算法有多种不同版本的变异机制,可表示为 DE/a/b 的形式。其中,a 表示变异操作基的类型,包括 rand 和 best 两种(rand 表示随机选择一个个体作为变异操作基,best 表示选择当前最优个体作为变异基);b 表示变异时差分项目的数量。最常用的变异机制为 DE/rand/1,即 $t_i(g) = x_{r1}(g) + F \times [x_{r2}(g) - x_{r3}(g)]$,其中 F 为取值在 0~1 的缩放比例因子。

3)交叉操作 在进行变异操作后,以一定的概率,使用目标个体 t_i 的部分变量来替换当前个体 x_i 的相应部分,得到测试个体 v_i。交叉方式有二项式交叉和指数交叉两种形式。

4)选择操作 从测试个体 v_i 和当前个体 x_i 中,选出较好的一个进入下一代搜索,即

$$x_i(g+1) = \begin{cases} t_i(g) & f[t_i(g)] < f[x_i(g)] \\ x_i(g) & \text{其他} \end{cases}$$

人们对经典差分进化算法提出了许多改进方案,主要集中在参数的自适应调整[16~20]和与其他算法组成混合算法[14~21]两个方面。值得一提的是,实验结果表明,就组合测试用例生成问题而言,这些改进算法均不如经典算法好。笔者试验了与进化代数相关的参数调节、与优化停滞代数相关的参数调节、与和声搜索算法组成混合算法、每隔一定代数对适应度较低的 20% 的个体进行激变等改进策略,每种算法独立运行 20 次,对比各算法得到的用例数量平均数和方差来判断算法的优劣。

2.2 数据存储结构

定义 4 组态。软件 t 个输入项的全部组合构成的集合称为强度为 t 的组态。假设一个系统共有四个输入项 I_1, I_2, I_3, I_4,则它的强度为 3 的组态为{(I_1, I_2, I_3),(I_1, I_2, I_4),(I_1, I_3, I_4),(I_2, I_3, I_4)}。

定义 5 组态表。对于组态中的各个元素,覆盖全部可选值的组合构成的集合称为组态表。假设输入项 I_1、I_2 均有两个可选值 0 和 1,则组态元素(I_1, I_2)的覆盖全部可选值的集合为{(0,0),(0,1),(1,0),(1,1)}。

a)可选值的存储,用链表存储各个输入项的可选值。使用这种存储方式,可以在可选值与其在链表中的位置之间建立一一对应关系。在优化过程中,设定各个输入项先后顺序的基础上,可以使用可选值在链表中的位置值来替代可选值的具体值,以解决 DE 中解的实数编码问题。

假设系统 S 有三个输入项,人为规定其顺序为 I_1, I_2, I_3。其中:I_1 有两个可选值 V_1 = {true, false};I_2 有三个可选值 V_2 = {0, 10, 20};I_3 有两个可选值 V_3 = {"PRC","USA"}。则测试用例{false,20,"PRC"}的编码为{1,2,0},如图 1 所示。

b)组态的存储,按照规定的顺序将系统的各个输入项存储在链表 L_i 中;以链表的形式将组态中各个输入项在 L_i 中的位置存储起来,就构成了组态的存储方式。系统 S 的组态存储结构如图 2 所示。

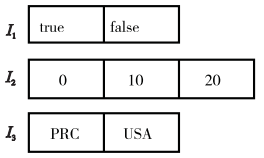


图1 可选值存储结构

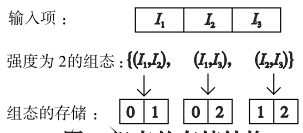


图2 组态的存储结构

c)组态表的存储,使用字典结构来存储组态表,以便通过组态来查询组态表。字典的 key 值为组态中的元素,如(0,1)(1,2)等;value 为组态表中该元素对应的组合构成的链表。系统 S 中,key = (0,1)时,value = {(0,0),(0,1),(0,2),(1,0),(1,1),(1,2)}。

2.3 组合覆盖评价方案

通过测试用例覆盖的有效组态元素的数量来评价该测试用例的优劣。假设现有测试用例集 $TR(i)$ 中包含的测试用例数量为 m,组态表中被这 m 个测试用例覆盖的部分为 C,未被覆盖的部分为 UC,测试用例 TC_j 能够覆盖 UC 中的 n 个,则 TC_j 的适应度为 n。

2.4 测试用例集生成流程

组合测试用例集的生成采用逐条扩展的一维扩展策略,即所谓的 one-test-at-a-time 策略。使用差分进化算法生成单个测试用例,加入测试用例集,直至组态表中的所有元素均被覆盖。生成过程分为初始化、随机生成 n 个测试用例、使用 DE 逐条生成其他测试用例三步。初始化部分主要完成配置参数及数据文件的读取、根据数据文件初始化组态链表、生成组态表字典等工作。当测试用例集 TR 中的用例数量较少时,由于组态表 TCon 中存在大量未被覆盖的元素,此时采用优化搜索和随机生成的方式生成的个体适应度几乎没有差别。为了提高效率,在进行优化生成之前,先随机生成 n 个测试用例存入 TR,并从 TCon 中删除被这 n 个测试用例覆盖的元素,对 4^{10} 组合强度为 3 的问题的实验表明,当 n 取组态数的 0.5% 比较合适。使用 DE 逐条生成其他测试用例时,首先随机生成 N 个个体组成初始种群;然后按照标准差分进化算法的基本步骤进行优化搜索;最后将找到的最优解加入 TR,并从 TCon 中删除被最优个体代表的测试用例覆盖的元素;如此循环,直至 TCon 为空。

3 实验与分析

3.1 实验数据表述方式

定义组合测试用例数据生成主要关注三个参数:输入项、输入项的可选值数和组合强度。假设待测系统有 m 个可选值数量为 n 的输入项,则其组合强度为 t 的组合测试数据生成问题可以描述为 $n^m - t$。比如,某待测系统包含三个可选值数量为 5 的输入项、两个可选值数量为 3 的输入项和四个可选值数量为 10 的输入项,求其组合强度为 3 的组合测试用例集的问题可以描述为 $5^3 3^2 10^4 - 3$。

3.2 DE/rand/1 与 DE/best/1 变异机制的性能比较

标准 DE 常用的变异方式有 DE/rand/1 和 DE/best/1 两种,通过实验对比了两种变异机制在解决组合测试数据生成问题时的效果。实验中,使用两种不同变异方式求解同一个问题,每种方式独立运行 10 次,通过对比其求得的测试用例数量的平均值来判断变异方式性能的优劣。实验对比结果如表 1

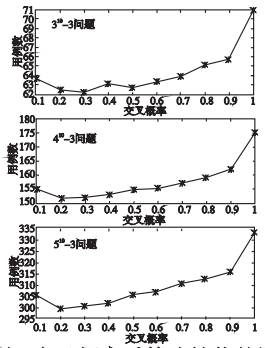
所示。由表 1 不难看出,对于组合测试用例集生成问题而言,随机变异基变异方式优于最优变异基变异方式。

表 1　变异方式对比结果

问题	变异方式	平均值
$4^{10}-2$	best	29.5
	rand	29.4
$4^{10}-3$	best	145
	rand	143.3
$4^{10}-4$	best	664.75
	rand	660.5
$2^{10}-5$	best	66
	rand	64.6
$2^{10}-6$	best	118
	rand	115.6

3.3　算法参数对性能的影响

差分进化算法的参数包括比例因子、交叉概率、种群大小、优化代数等。由于算法具有较强的参数敏感性,为了寻找较好的参数配置方案,设计了对比实验来考察这些参数对算法性能的影响。实验中,选定一个参数 A,保持其他参数不变,为 A 赋值 a_1,计算 10 次,求得测试用例数量的平均值$\overline{A_{a1}}$;再为 A 赋值 a_2,计算 10 次,求得测试用例数量的平均值$\overline{A_{a2}}$;持续这一过程,直至对参数 A 的实验结束。实验结果如图 3 ~ 6 所示。



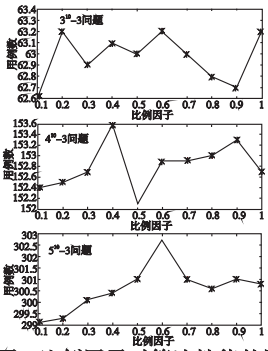
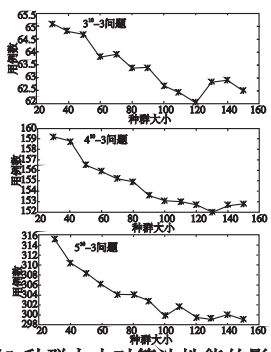


图3 交叉概率对算法性能的影响　图4 比例因子对算法性能的影响

由图 3 可以看出,算法生成的测试用例的数量呈现先抑后扬的变化趋势,最优点一般出现在 0.2 ~ 0.3 附近。

由图 4 可以看出,比例因子对算法的影响存在较大的不确定性,相对最优解多出现在 0.1 处。



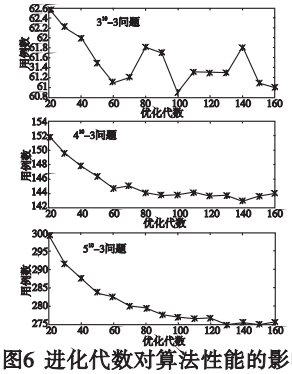


图5 种群大小对算法性能的影响　图6 进化代数对算法性能的影响

由图 5 可以看出,算法生成的测试用例数量随种群的增大而减小,但是当种群大小大于 100 时,算法找到的测试用例数量的变化率趋于平缓。

由图 6 可以看出,算法生成的测试用例数量随进化代数的增大而减小,但是当进化代数大于 60 时,算法找到的测试用例数量的变化率趋于平缓。

需要指出的是,上述实验仅考虑了各参数对算法的独立影响,没有考虑各参数之间的交互作用,而且仅考察了 $3^{10}-3$、$4^{10}-3$、$5^{10}-3$ 三类问题,实验样本还不够丰富。

3.4　与其他方法的比较

为了检验方案的性能,将其与美国国家标准与技术研究院(National Institute of Standards and Technology,NIST)提供的几种常用组合测试用例生成方案做了对比实验,实验结果如表 2 所示。

表 2　测试用例数量对比

问题	组合数	测试用例数量			
		IPOG	IPOG-F	PICT	DE
$4^{10}-2$	720	30	29	31	28
$4^{10}-3$	7 680	150	157	160	140
$4^{10}-4$	53 760	695	731	742	658
$4^{10}-5$	258 048	3 079	3 162	3 161	2 814
$4^{10}-6$	860 160	12 778	12 727	12 473	12 027

由表 2 可以看出,不论组合数为多少,差分进化算法均能够生成更少的测试用例。当组合数较少时,差分进化算法与其他三种算法生成的测试用例数量差别不大,随着组合数的增加,数量的差别越来越大。

4　结束语

基于 one-test-at-a-time 策略,使用差分进化算法求解组合测试用例生成问题的方案能够有效减少生成的测试用例的数量,从而提高测试效率。实验结果证明了这种方案与其他经典方案相比,具有较大的优势。由于每条测试用例均由差分进化算法优化搜索得到,所以当组合数较多时,测试用例生成的过程比较耗时。不过组合数越多,方案在生成的测试用例数量方面的优势就越明显;考虑到测试中测试用例的准备、执行、测试结果记录等过程也是比较繁琐和耗时的,因而测试前的时间投入具有较高的回报率。

参考文献:

[1] 严俊,张健. 组合测试:原理与方法[J]. 软件学报,2009,20(6):1348-1405.

[2] 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,2012,34(12):2532-2541.

[3] 黄陇,杨宇航,李虎. 参数配对及 n-way 组合覆盖算法研究[J]. 计算机学报,2012,35(2):257-269.

[4] 史亮,聂长海,徐宝文. 基于解空间树的组合测试数据生成[J]. 计算机学报,2006,29(6):849-853.

[5] SONG C, PORTER A, FOSTER J S. iTree: efficiently discovering high-coverage configurations using interaction trees[C]//Proc of International Conference on Software Engineering. 2012: 903-913.

[6] 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.

[7] 查日军, 张德平, 聂长海. 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.

[8] 梁亚澜,聂长海. 覆盖表生成的遗传算法配置参数优化[J]. 计算机学报,2012,35(7):1522-1538.

[9] McCAFFREY J D. Generation of pairwise test sets using a genetic algorithm[C]// Proc of COMPSAC. 2009: 626 - 631.

[10] 贺毅朝,王熙照,刘坤起,等. 差分演化的收敛性分析与算法改进[J]. 软件学报, 2010,21(5):875- 885.

[11] KUHN D R, REILLY M J. An investigation of the applicability of design of experiments to software testing[C]//Proc of the 27th Annual NASA/IEEE Software Engineering Workshop. Washington DC: IEEE Computer Society, 2002:91-95.

(下转第 1455 页)

等。对状态图使用状态约简算法并将顺序图和状态图转换为 PROMELA 代码,使用 SPIN 验证工具对顺序图和状态图语义一致性进行检测,得到检测结果如图 7 所示。图 7 左侧和右侧分别是两个顺序图与状态图一致性检测结果,结果表明,左侧顺序图与状态图是一致的;对右侧顺序图与状态图,工具执行到 18 个状态和 18 个迁移的时候由于阻塞停止运行,说明两者是不一致的。

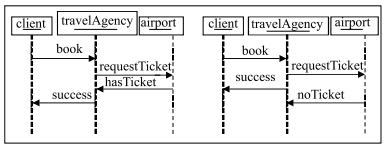


图 5　订票系统顺序图

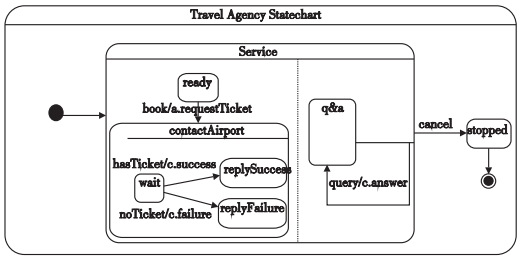


图6　订票系统中旅行社状态图

```
state ¬vector 24 byte, depth reached37,errors: 0
    27 states, stored
     2 states, matched
    29 transitions (= stored +matched )
     0 atomic steps
hash conflicts: 0 (resolved )

stats on memory usage (in Megabytes):
    0.001    equivalent memory usage for states
(stored *(State ¬vector + overhead ))
    0.290    actual memory usage for states
   64.000    memory used for hash table (¬w24 )
    0.343    memory used for DFS stack(¬m10000 )
   64.539    total actual memory usage
```

```
state ¬vector 24 byte, depth reached  24, errors: 1
    18 states, stored
     0 states, matched
    18 transitions (= stored +matched )
     0 atomic steps
hash conflicts: 0 (resolved )

stats on memory usage (in Megabytes ):
    0.001    equivalent memory usage for states
(stored *(State ¬vector + overhead ))
    0.290    actual memory usage for states
   64.000    memory used for hash table (¬w24)
    0.343    memory used for DFS stack(¬m10000 )
   64.539    total actual memory usage
```

图7　订票系统一致性检测结果

另外,为了验证改进的方法对状态数量的优化,本文对文献[3]中的 ATM 系统、文献[5]中的订票系统、文献[6]中的安全系统、文献[11]中的电话呼叫系统和集成设计平台实际应用中的攻防系统的一致性进行验证,并将验证过程中产生的状态数量进行对比。图 8 和 9 给出了验证过程中复杂度情况,可以看出使用文中状态约简和转换方法后检测工具 SPIN 中产生的状态和迁移数量较少,并且随着模型复杂度的提高,检测效率的提高更为明显。

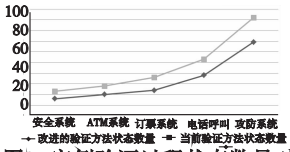


图8　实例验证过程状态数量对比

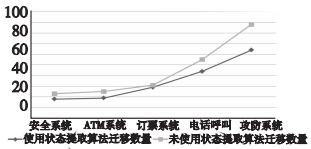


图9　实例验证过程迁移数量对比

4　结束语

本文给出了一种改进的顺序图和状态图一致性检测方法,并实例验了证该方法的可行性。该方法有如下优点:a)给出基于消息序列的顺序图语义和基于迁移序列的状态图语义,并对顺序图和状态图的语义一致性进行精确描述;b)提出了状态约简算法,减少检测过程中冗余状态和迁移;c)给出了改进的顺序图和状态图到 PROMELA 的转换方法,将顺序图和状态图分别转换成一个进程,转换后代码结构简单、执行效率高。后续工作包括对其他 UML 模型如对象图、协作图、活动图之间的一致性研究,给出检测出的一致性反例到 UML 模型的映射方案[12]等。

参考文献:

[1] ZHANG Shao-jie, LIU Yang. An automatic approach to model checking UML state machines[C]//Proc of the 4th International Conference on Secure Software Integration and Reliability Improvement Companion. [S.l.]:IEEE Press, 2010:1-6.

[2] CHENG Liang, ZHANG Yang. Model checking security policy model using both UML static and dynamic diagrams[C]//Proc of the 4th International Conference on Security of Information and Networks. New York:ACM Press, 2011:159-166.

[3] TIMM S, KNAPP A, MERZ S. Model checking UML state machines and collaborations[J]. Electronic Notes in Theoretical Computer Science, 2001, 55(3):357-369.

[4] MMIN H S, CHUNG S M, CHOI J Y. Deriving system behavior from UML state machine diagram: applied to missile project[J]. Journal of Universal Computer Science, 2013, 19(1):53-77.

[5] ZHAO Xiang-peng, LONG Quan, QIU Zong-yan. Model checking dynamic UML consistency[C] // Proc of the 18th International Conference on Formal Methods and Software Engineering. Berlin: Springer, 2006: 440-459.

[6] ORNA G, MELLER Y, YORAV K. Applying software model checking techniques for behavioral UML models[C]// Proc of the 18th International Symposium on Formal Methods. Berlin: Springer, 2012: 277-292.

[7] UML 2.0 superstructure specification[EB/OL]. http://www.omg.org/spec/UML/2.0/.

[8] 古思山,蔡树彬,李师贤. 一种 UML2 的交互的形式化语义[J]. 计算机科学与探索, 2012, 6(7): 631-643.

[9] RUSS M, HAMILTON K. Learning UML 2.0[M]. [S.l.]:O'Reilly Media, 2008.

[10] HOLZMANN G. SPIN model checker: the primer and reference manual[M]. Boston: Addison-Wesley Professional, 2004.

[11] 李兴锋,张新常,杨美红,等. 基于 SPIN 的模块化模型检测方法研究[J]. 电子与信息学报, 2011, 33(4) : 902-907.

[12] ISLAM A, SCHNEIDER S, TREHARNE H. Towards a practical approach to check UML/fUML models consistency using CSP[C]//Proc of the 13th International Conference on Formal Methods and Software Engineering. Berlin :Springer-Verlag, 2011: 33-48.

(上接第 1451 页)

[12] 覃晖,周建中,王光谦. 基于多目标差分进化算法的水库多目标防洪调度研究[J]. 水利学报,2009,40(5): 513-519.

[13] 刘波,王凌,金以慧. 差分进化算法研究进展[J]. 控制与决策, 2007,22(7):721-729.

[14] 王凌,钱斌. 混合差分进化与调度算法[M]. 北京:清华大学出版社, 2012:33-48.

[15] 孟红云,张小华,刘三阳. 用于约束多目标优化问题的双群体差分进化算法[J]. 计算机学报,2008,31(2):228-235.

[16] 肖术骏,朱学峰. 一种改进的快速高效的差分进化算法[J]. 合肥工业大学学报,2009,32(11):1700-1703.

[17] 张雪霞, 陈维荣, 戴朝华. 带局部搜索的动态多群体自适应差分进化算法及函数优化[J]. 电子学报,2008,38(8):1825-1830.

[18] 戈剑武,祁荣宾,钱锋,等. 一种改进的自适应差分进化算法[J]. 华东理工大学学报,2009, 35(4):600-605.

[19] PAN Quan-ke, SUGANTHAN P N, WANG Ling, *et al*. A differential evolution algorithm with self-adapting strategy and control parameters [J]. Computers & Operations Research, 2011, 38(1):394-408.

[20] MALLPEDDI R, SUGANTHAN P N, PAN Quan-ke, *et al*. Differential evolution algorithm with ensemble of parameters and mutation strategies [J]. Applied Soft Computing, 2011, 11(2):1679-1696.

[21] LIAO T W. Two hybrid differential evolution algorithms for engineering design optimization[J]. Applied Soft Computing, 2010, 10(4) : 1188-1199.