基于遗传算法的动态可变参数的测试数据自动生成工具

史娇娇 姜淑娟
(中国矿业大学计算机科学与技术学院 徐州 221116)

摘 要 测试数据的生成是实现软件测试自动化的关键,这一技术的实现大大节省了软件开发的时间和费用。利用遗传算法的理论与算法特点,建立了动态可变参数的测试数据自动生成工具。通过该工具的可视化界面可以动态地输入遗传算法参数,而且能够根据不同的路径选择输入相应的适应度函数,克服了以往在源代码中修改适应度函数的缺陷。最后通过两个实验,证明了算法的优越性。

关键词 软件测试,测试数据自动生成,可视化,遗传算法,适应度函数
中图法分类号 TP311 **文献标识码** A

Automatic Test Data Generation Tool of Dynamic Variable Parameters Based on Genetic Algorithm

SHI Jiao-jiao JIANG Shu-juan
(College of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China)

Abstract Test data generation is the key of implementing software test automation, and the realization of this technology can greatly save for time and money of software development. This paper, using genetic algorithm theory and algorithm characteristics, established a test data automatic generation tool of variable parameter dynamic. Through the visual interface of the tool, you can input genetic algorithm parameters dynamically. And according to the different path selection you can input corresponding fitness function, overcome defects that fitness function is modified in the source code in the past. Finally through the two experiments, we can prove the superiority of the algorithm in this paper.

Keywords Software testing, Automatic test data generation, Visual, Genetic algorithm, Fitness function

1 引言

软件测试作为保证软件质量的重要手段,使用各种算法自动产生测试用例以提高测试的自动化,成为近年来研究的热点。遗传算法作为一种启发式的搜索寻优算法,将测试数据的生成转化为求解一组优化数据的方法,这种算法的关键在于适应度函数的设计。

近年来,国内外对于将遗传算法应用于测试数据自动生成的方法已有一些研究。Wegener 等[8]的实验表明,为使程序达到分支覆盖,遗传算法生成的测试数据数比随机法小一到两个数量级。James Miller 等[6]针对布尔型和枚举型的变量,提出利用程序依赖图和遗传算法自动生成测试数据。汪浩等[2]给出了遗传算法的形式化的表示和一个基于此算法的测试数据生成系统原型;钱肖英等[1]将遗传算法应用到特定背景,提出了包括数值型、非数值型和类对象软件测试数据的自动生成方法,并得出遗传算法比爬山法和随机法生成测试数据的效率都高的结论。王捷民等[4]对基于改进的自适应遗传算法的测试数据自动生成进行研究,针对软件测试数据的自动生成,提出了一种自适应遗传算法和爬山算法相结合的改进算法 HCGA。巩敦卫等[11]将被测程序表示成一棵二叉树,采用赫夫曼编码方法将目标路径表示成二进制串,结合遗传算法提出了一种新的多路径覆盖的测试数据生成方法。白凯等[12]结合遗传算法设计了一个测试数据的自动生成模型,实现了单元测试中路径分支覆盖测试数据的自动生成。

但是,在目前的测试数据模型中,适应度函数都在源代码中设计执行。当需要测试不同的路径时,必须在源代码中更改相应的适应度函数,这就使得现有的测试数据工具的使用率大大降低。针对这一问题,提出一种基于遗传算法的测试数据生成算法,建立了动态可变参数的测试数据自动生成工具,它能够在其可视化界面中,动态地输入遗传算法参数,特别是能够根据不同的路径选择输入相应的适应度函数,节省了大量的时间。

2 基于遗传算法的测试数据自动生成模型

将遗传算法运用到测试数据自动生成,主要是利用评价函数将遗传算法与具体应用问题连接起来,它的构造直接影响问题求解的效率。这里首先对被测程序进行静态分析,获得程序分支路径集合,并对各谓词条件进行程序插桩;根据选择的目标路径,在可视化界面上输入相应的遗传算法参数及适应度函数,利用遗传算法的搜索策略来生成测试数据。基

到稿日期:2011-11-11 返修日期:2012-02-17 本文受国家自然科学基金(60970032),江苏省自然科学基金(BK2008124),中国矿业大学科学研究基金(OD080310)资助。

史娇娇(1988-),女,硕士生,主要研究方向为软件测试、测试数据,E-mail:sjiao888@126.com;**姜淑娟**(1966-),女,博士,教授,博士生导师,CCF 会员,主要研究方向为软件工程、软件测试、程序设计语言等。

于遗传算法的测试数据自动生成模型如图 1 所示。该模型包含测试环境构造、遗传算法和测试运行环境 3 部分。

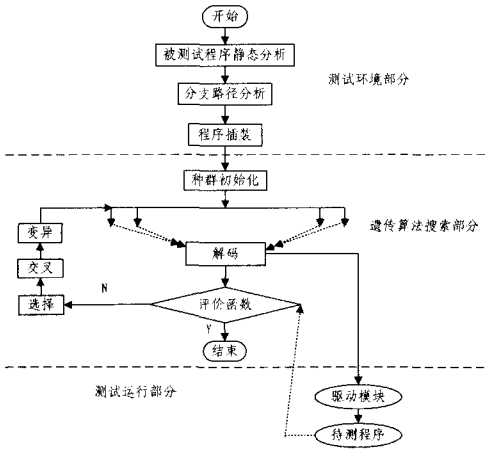


图 1　基于遗传算法的测试数据自动生成模型

第一部分是整个系统的基础，它除了获取程序中有用的参数外，还对程序进行插装等操作来构造相应的测试运行环境。

第二部分即遗传算法部分是系统的核心，它主要按照第一部分生成的编码参数格式，利用构造函数及克隆技术构造相应的染色体串，用可变长数组生成初始种群，节省了大量的内存空间。通过对该种群进行反复的选择（如果选择种群数量大于种群规定数量，则淘汰适应值最小的染色体；如果选择种群数量小于种群规定数量，则加入适应值最大的染色体。）、交叉（交叉后的个体首先调用解码函数，再判断解码后的参数是否在范围内，若不在就重复交叉操作。同时对交叉后的个体进行变异操作，并记录交叉位和选择交叉的个体。）和变异操作，来引导种群不断地向目标值进化，直到最终找到解或达到限定的运行代数。

第三部分是测试运行环境。第二部分遗传算法需要对种群中每一个个体的优劣进行评估。其主要是根据插装后的被测程序，通过实时地调用测试系统来返回适应度值，根据评判标准决定程序的运行与终止，以此来指导遗传算法的搜索。

3　动态可变参数的可视化输入

3.1　适应度函数构造

适应度函数是遗传算法与具体应用问题的唯一接口，是种群中个体优劣的一种量化反映。这里采用基于路径覆盖的测试数据生成，将目标函数的求解转化为求解一组优化的测试数据。

假设目标路径由一串节点序列组成，其中一些节点关联相应的分支判断，即一个逻辑表达式，其值可以取真或取假。在执行过程中，为了能覆盖指定路径，需要找到一组适当的输入参数来满足沿途相关的分支判断，使得相关节点的有效分支被执行，所有的这些条件可以被整合成一个单函数。

Korel 提出了“分支函数叠加法”[7]，其依据程序插装技术，将简单关系表达式（E1 op E2，其中 E1 和 E2 是算术表达式，op 为关系运算符）转换成等价形式 F rel 0，构造方法需要检测指定路径的沿途条件以判断语句的真假值关系，表 1 给出了一些常用的分支函数。当分支谓词为假时，分支函数为正；当分支谓词为真时，分支函数为负。要使某条路径被覆盖，则该路径上的所有分支谓词应取真值，即分支函数应为负。由于分支函数直接构成了适应度函数，不能为负值，因此将分支函数修改为当分支谓词为真时，分支函数取 0；当分支谓词取假时，分支函数依然取真值。因此如果某条路径的所有分支函数都为 0 时，表示该路径被全部覆盖。

表 1　分支函数构造

分支谓词表达式	适应函数值	
	分支函数 F	rel
E1>E2	E2−E1	<
E1>=E2	E2−E1	<=
E1<E2	E1−E2	<
E1<=E2	E1−E2	<=
E1==E2	abs(E1−E2)	==
E1!=E2	−abs(E1−E2)	<

本文在“分支函数叠加法”[7]的基础上进行改进，即针对程序中的谓词节点构造的分支函数的倒数进行叠加。分支函数是一个实值函数，也即分支谓词到实值的一个映射，因此可以把分支问题数量化，量化地描述在测试数据的驱动下，被测单元的实际执行路径对选定路径的覆盖程度。根据前面的讨论，将分支函数构造为表 1 所列形式，使适应度函数能够区分 E1<E2 和 E1<=E2 以及 E1>E2 和 E1>=E2 的情况。适应度函数为

$$fit(x)=\frac{1}{1+f(x_i)} \tag{1}$$

式中，$f(x_i)$为各分支函数的值。这样就使得适应度值 $fit(x)$ 的范围固定在(0,1]，当分支函数取 0 时，适应度值达到最大。这样做的含义是，可以把较小的变量看作当前生成测试数据的变量，较大的一个看作目标解。从 $f(x_i)$定义可以看出，当 E1 与 E2 越接近时，$f(x_i)$的值越小，当达到 0 时，适应度函数取最大值，表示已经满足目标解。利用整条路径分支函数极小化来趋近目标路径，将大大提高覆盖效率。

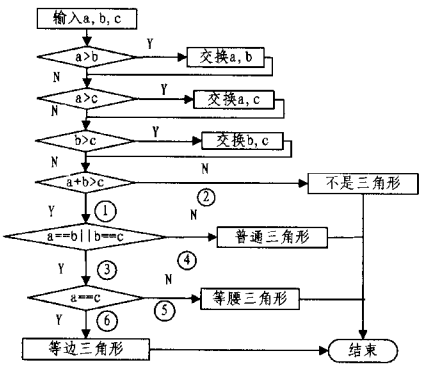
3.2　一个实例

遗传算法设计的关键在于适应度函数的构造，这里将利用一个具体的测试实例说明其构造方法。判定三角形类别的程序在众多软件测试研究中被作为一个基准程序来使用，很多学者都使用它来检测路经测试覆盖方法。不失一般性，本文同样将其作为被测程序，3 个整数用作三角形的 3 条边作为输入，然后判定输出为哪一类三角形：非三角形、普通、等腰或等边三角形。图 2(a)和图 2(b)分别给出了三角形分类问题的代码片段和程序流程图。根据概论知识，路径“①-③-⑥”是通过测试数据随机生成法比较难达到覆盖的，因为即使将一个较大范围的整数作为输入，也只有少量的输入能满足该条分支。假设该测试单元的参数是以 7 位为倍数的二进制编码，则 a,b,c 至少有 $2^7*2^7*2^7=2097152$ 种组合，其中能够覆盖路径“①-③-⑥”的不超过 10 组，所以选取这条路径作为被测路径，能充分验证算法的搜索能力。

将插装语句放在紧挨分支判断的前面，使得分支判断语句能够在插装语句执行后立即执行，而插装语句也能够准确地获取分支判断的状态。图 2(a)中的 7、9、11 是插装的分支函数，而适应度函数 F 由可视化界面动态输入，根据选择的路径不同，输入的适应度函数也不同。

```
1 double type(int a,int b,int c)
2 { string type;
3   double F;
4   if(a>b) change(a,b);
5   if(a>c) change(a,c);
6   if(b>c) change(b,c);
7   f1=c-(a+b) ;
8   if(a+b>c)
9     { f2=min(abs(a-b),abs(b-c));
10      if(a==b||b==c)
11      { f3=abs(a-c);
12        if(a==c) type="等边三角形";
13      else type="等腰三角形";
14    }else type="普通三角形";
15  } else type="不是三角形";
16 if(f1<0) { f1=0;}
17 return F; } //F为适应度函数,由可视化界面输入,可根据目标路径,选择输入
```

(a) 三角形程序部分代码



(b) 三角形程序流程图

图 2

3.3 可视化实现

本工具由 Java 语言编写,利用 GroupLayout 布局实现对程序中遗传算法参数的动态输入,特别是对适应度函数的动态输入。基本遗传算法参数:种群数量、遗传代数、变异概率、交叉概率和染色体长度均可在可视化界面中修改,通过中间变量,从文本框中读取各参数,实现界面与程序中变量的传递。设计输入适应度函数,首先要对文本框中输入的函数进行字符串解析,将解析后的字符串传递到 FunctionFitness 类的适应度计算方法中,然后调用被测程序,完成对可视化界面中适应度函数的传递。

适应度函数输入时,可根据路径选择情况的不同,在可视化界面上根据需要动态地输入参数,其克服了以往在源代码中修改适应度函数的缺陷,增加了测试工具的灵活性及实用性。在具体应用时,为了直观地观察适应度函数的变化,这里将适应度值的变化范围同比例放大 100 倍,这样可以看到当覆盖目标路径时,适应度值 100 为适应度函数最大值,即达到最优解。例如,设置遗传算法参数:运行次数为 2,种群数量为 80,遗传代数为 150,变异、交叉概率分别为 0.02、0.5,染色体长度为 21,输出的是等边三角形(适应度函数一栏处为 100(1/(1+$f1$)+1/(1+$f2$)+1/(1+$f3$))/3);然后将上述参数进行适当修改,输出为等腰三角形(100(1/(1+$f1$)+1/(1+$f2$))/2),图 3 和图 4 分别说明了这两种情况。

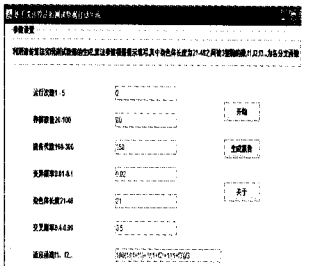


图 3 输出等边三角形的设计

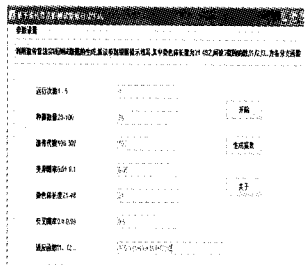


图 4 输出等腰三角形的设计

4 实验分析

4.1 判定三角形程序

鉴于以上对三角形程序的设计,为了验证本文方法的性能,这里给出了能够实现生成全部分支覆盖测试数据的模型,以生成等边三角形为例进行说明。

根据测试工具的可视化界面提示,遗传算法的最优参数范围如图 5 参数设置部分所示。本文采用二进制编码,被测程序有 3 个输入参数,遗传编码为 3 参数级联编码,编码长度共 21 位,每个参数 7 位,种群规模大小初始化为 M=50,最大世代数 150,运行次数 1 次,交叉概率0.6,变异概率0.05。另外,本实验设定算法终止条件为满足以下两个条件之一:

- 找到最优解,即适应度为 100 的解;
- 达到最大进化代数,当程序进化满 150 代后,不管有没有找到最优解都将退出。

(1)输入基本参数后,选择输出等边三角形,在适应度函数处输入 100(1/(1+$f1$)+1/(1+$f2$)+1/(1+$f3$))/3,点击开始,出现如图 5 所示的界面,生成的部分报告如图 6 所示。从图中可以看出,在第 25 代时找到最优解,个体 1 为最优个体,适应度为 100,对应的变量为 26,26,26。

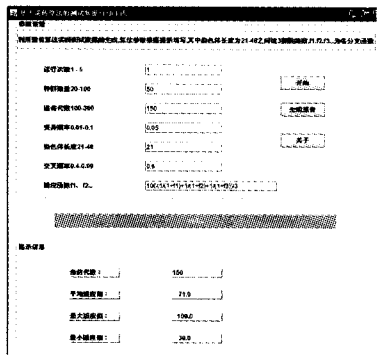


图 5 程序运行图

```
第1/1次运行,当前代为25代,共150代
--------------------------------------------------
                模拟计算统计报告
世代数  25
个体  染色体编码          适应度值  对应变量  父个体交叉位置
1>  010001101000100100010  100.0  26 26 26 (15, 6)   0
2>  010001101000100100010  100.0  26 26 26 (21, 7)  14
3>  010001101000100100001  83.3   26 26 25 (25,27)  19
4>  010001101000100011111  73.3   26 26 22 (40, 4)   2
5>  010001101000100011011  70.7   26 26 19 ( 3,11)   3
6>  010001101000100010101  69.3   26 26 15 (29,24)   0
7>  010001101000110000111  68.0   26 26  4 (47,48)  15
8>  010110101000100100011  70.3   26 26 18 (16,30)   0
9>  111111101100100010100  68.3   26 26 44 ( 9,17)   0
10> 110001111000100001110  67.0   76 76  6 ( 5, 2)   7
```

图 6 生成报告

(2)将第 1 代、第 12 代、第 25 代部分个体的适应度及相关参数整理如表 2—表 4 所列。

表 2　第 1 代部分个体

个体编码	染色体编码	参数值			适应度
		A	B	C	
1	01011010100010010011	34	26	26	70.3
2	11000011010110101110	75	66	66	70.0
3	11000101100110010001	76	79	31	45.0
4	01100000111000000001	37	43	6	41.3
5	10011011010110010001	59	66	25	40.7
6	00111011001101010111	21	59	41	38.3
7	01011010100010010011	26	10	34	35.3
8	11010000000010001111000	81	3	43	37.7
9	11011101011001001011O	85	68	16	38.3
10	10011010111010100100l	59	44	56	39.0

表 3　第 12 代部分个体

个体编码	染色体编码	参数值			适应度
		A	B	C	
1	01000110100001001000000	26	26	25	83.3
2	01000110100001000111111	26	26	22	73.3
3	01000110100001000110010	26	26	19	70.7
4	11000011010110101010110	75	66	66	70.0
5	01000110100001100101111	26	26	16	69.7
6	00111011001101010111l	26	26	41	68.7
7	11000101100110010100l	76	79	31	45.0
8	01100000111000000010O1	37	43	6	41.3
9	01001110100010011011l	29	26	41	45.3
10	01000110100000010100O	26	25	31	53.7

表 4　第 25 代部分个体

个体编码	染色体编码	参数值			适应度
		A	B	C	
1	01000110100001001000010	26	26	26	100
2	01000110100001001000010	26	26	26	100
3	01000110100001001000000	26	26	25	83.3
4	01000110100001000111111	26	26	22	73.3
5	01000110100001000110010	26	26	19	70.7
6	01000110100011010111110	26	26	47	68.0
7	01011010100010010001l	34	26	26	70.3
8	01011010100010010001l	29	29	13	68.3
9	01000110100010001010l	26	26	15	69.3
10	11000111100010000111O	76	76	10	67.0

由以上 3 张表可知，随着进化过程的继续，群体的总体适应度在增加，说明算法正朝着最优解的方向收敛，当程序运行到 25 代时，适应度函数为 100，即找到最优解。对每一代个体的适应度进行分析比较，发现每一代个体的最大适应度与种群的进化代数存在图 7 所示的关系，说明遗传算法在程序测试数据自动生成中能逐渐改善个体，使其越来越接近的目标路径，最终达到所设定的目标路径。

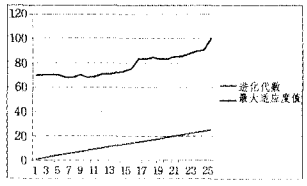


图 7　进化代数与最大适应度值之间的关系

(3)为了测试本文算法的优劣，现分别使用该算法和文献[12]中的算法(记作 w 方法)选择三角形分类程序中的 4 条可行路径，设定找到输出等边三角形的路径为结束条件。实验时记录每次找到目标路径时的进化代数，针对不同数据范围进行 100 次搜索，分别分析这两种算法的进化代数与运行时间。由于每次运行的进化时间只有几 ms，因此记录了 100 次的总进化时间。两种算法的对比结果如表 5 所列。

表 5　两种算法运行结果

数据范围	种群规模	平均进化代数		总进化时间(s)	
		本文方法	w 方法	本文方法	w 方法
$[0,100]^3$	30	23.5	62.7	0.163	0.324
$[0,200]^3$	50	36.4	73.5	0.185	0.502
$[0,500]^3$	80	45.8	205.8	0.689	3.875
$[0,1000]^3$	100	70.2	413.6	1.325	6.974

从表 5 可以看出：

①随着输入数据范围的增大，两种方法找到目标路径的进化代数和进化时间增加，这是由于输入数据范围的增大使得搜索难度加大；

②随着输入数据范围的增大，时间差距越来越大，说明本文方法在大范围输入下优越性更明显。

对于一个目标路径的测试数据的进化代数，以输入范围 $[0,100]^3$ 为例，记录了运行 100 次覆盖目标路径的测试数据的进化代数，并计算其平均值，如表 6 所列。

表 6　覆盖目标路径的测试数据的平均进化代数

	非三角形	普通三角形	等腰三角形	等边三角形
本文方法	1	1	4.8	25.6
w 方法	1	1	5.9	285.4

从表 6 可以看出：

①覆盖非三角形和普通三角形路径的测试数据容易找到，两种方法均在第 1 代找到这些数据；

②覆盖等腰三角形的测试数据也在较少的进化代数下找到，两个方法差别不大；

③覆盖等边三角形的测试数据难以寻找，本文方法平均需要 25.6 代可找到，而 w 方法平均需要 285.4 代。

由此可以看出，对于寻找难以覆盖路径的测试数据，本文方法在进化代数方面明显优于 w 方法。

4.2　冒泡排序程序

为了进一步验证本文方法的性能，实验除了选择三角形分类程序外，还对冒泡排序程序进行分析验证。根据文中适应度函数构造方法，在可视化界面上动态输入基本遗传算法参数和适应度函数，并选择其中 5 个可执行路径进行算法验证。以输入范围 $[0,100]^3$ 为例，种群规模为 50，最大运行代数为 150，计算找到每个目标路径的平均进化代数和 100 次的总进化时间。结果如表 7 所列。

表 7　冒泡排序程序平均进化代数和总进化时间

	路径 1	路径 2	路径 3	路径 4	路径 5	总进化时间(s)
本文方法	1.4	2.5	3.5	7.4	18.6	0.168
w 方法	1.8	2.9	4.8	24.5	29.8	0.463

从表 7 可以看出，对于复杂路径，本文方法在进化代数和进化时间方面的性能都优于 w 方法。

结束语　本文实现了基于遗传算法的动态可变参数的测试数据自动生成工具，及在可视化界面中动态输入遗传算法参数，特别是对适应度函数的可视化输入，克服了以往针对不同路径选择在源代码中修改适应度函数的缺陷，并通过两个实验验证了该工具的可行性及实用性。

参 考 文 献

[1]　钱肖英. 基于遗传算法的测试数据自动生成方法的研究[D]. 杭州：浙江工商大学，2008：25-60

(下转第 155 页)

储法和现实数据的一致度，即将真实世界作为模型 M'，计算 M 与 M' 之间的一致度。下面分别采用多组数据并比较一致度的结果，如表 1 所列。

表 1　加聚解聚法与并行存储法输出结果对比表

序号	高分辨率实体	运行于高分辨率受到损伤	运行于低分辨率受到损伤	真实损伤	加聚解聚法输出结果	并行存储法输出结果	加聚解聚法一致度	并行存储法一致度
	M_1	10%	15%	25%	25%	25%		
1	M_2	10%	15%	25%	25%	25%	100%	100%
	M_3	10%	15%	25%	25%	25%		
	M_1	20%	30%	50%	30%	40%		
2	M_2	5%	20%	25%	30%	25%	55.3%	75.9%
	M_3	5%	10%	15%	30%	25%		
	M_1	50%	15%	55%	30%	60%		
3	M_2	5%	10%	15%	30%	15%	24.9%	83.8%
	M_3	5%	5%	10%	30%	15%		

并行存储法记录了高分辨率交互信息，因此能表现出与现实世界之间更高的一致性。当运行高分辨率层次时，各实体受到的损伤区别很大；运行于低分辨率层次时，受到的损伤区别较小，并行存储法的性能表现最优。图 5 表示了经历过加聚解聚过程的模型的一致度对比，紫线为并行存储方法，黄线为加聚解聚方法。

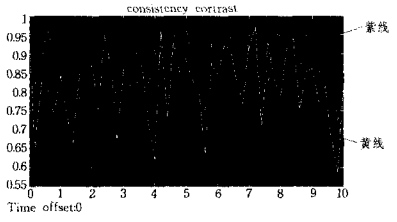


图 5　一致度分析结果对比图

在本次试验中，随机产生一个 3×3 矩阵，矩阵第一列为模型运行于高分辨率层次时高分辨率的 3 个实体分别受到的损伤，第二列为模型运行于低分辨率层次时 3 个实体分别受到的损伤，第三列为模型运行于高分辨率层次时 3 个实体分别受到的损伤。经过一次聚合与解聚，来对两个建模方法的仿真度进行比较，对比结果如表 1 所列。

4.2　在交互冲突上进行比较

加聚解聚法常用的解决交互冲突的方法为加锁机制。当 N 与 M 进行交互时，先将 M_1，M_2，M_3 中攻击能力的同态属性进行加锁，交互完成后才能解锁。此时首先需要将 M 解聚，然后计算 M_1，M_2，M_3 中哪些属性是攻击能力的同态属性，进而加锁，过程很清晰。解聚后还要重新回到低分辨率模型，一次加聚解聚和对同态属性的验证需要较长时间。并行存储方法采用验证机制，避免了重复更新。当 N 与 M_1 中的攻击能力交互时，更改 M_1 中的攻击能力，同时更改 M 中对 M_1，M_2，M_3 的攻击能力聚合后的信息。由于对 M 中该信息更改只是更改的虚拟值，因此当将该虚拟值的变化反馈到高分辨率层次中时才进行验证。如果需要更新便更新，否则弃掉此次变化并通知 N，这在很大程度上节省了解决冲突的时间。

结束语　在多分辨率仿真领域，数据的一致性问题一直是难以解决的问题，而如何能更高效地解决并发冲突问题，也作为一个课题被提出来。本文在已有成果的基础上提出了基于并行存储的多分辨率仿真模型，并提出了新的解决思路。当前，基于并行存储的多分辨率仿真也面临一些挑战。如何根据属性的特性来判断是否存储在高分辨率仿真中，这个标准不易实施。为了将高分辨率模型和低分辨率模型中的数据独立地存储，每个模型需要两个独立的存储器，如何处理两个存储器之间的联系，也有待进一步探索。

参 考 文 献

[1] 刘宝宏，黄柯棣. 多分辨率模型系中的一致性问题研究[J]. 系统仿真学报，2005，17(9)：2057-2059

[2] 尹华飞，吕品，等. 多分辨率仿真模型互联运行问题研究[J]. 中国体视学与图像分析，2010，15(3)：269-273

[3] 周华任，马亚平，等. 战争模拟多分辨率建模研究[J]. 系统仿真学报，2009，21(21)：6833-6836

[4] 朱松岩，江敬灼，等. 聚合解聚及其一致性问题研究[J]. 军事运筹与系统工程，2008，22(3)：33-38

[5] 刘宝宏. 多分辨率建模的理论与关键技术研究[D]. 长沙：国防科学技术大学，2003

[6] 周彦，戴剑伟. HLA 仿真程序设计[M]. 北京：电子工业出版社，2002，6

[7] Drewry D T, Reynolds P F, Emanuel W R. Optimization as a Tool for Consistency Maintenance in Multi-resolution Simulation[R]. University of Virginia, 2002

[8] Franceschini R W. Computational for Disaggregation[C]//Paper of the 9th CGF &BR Conference. 2000

[9] Natrajan A. Consistency Maintenance in Concurrent Representation [D]. USA: School of Engineering and Applied Science, The Univ. of Virginia, 2000

(上接第 127 页)

[2] 汪浩，谢军凯，高仲仪. 遗传算法及其在软件测试数据生成中的应用研究[J]. 计算机工程与应用，2001，15(12)：64-68

[3] 董敏，毕盛，齐德昱. 基于正则表达式的测试数据自动生成技术[J]. 计算机工程，2009，35(16)：29-31

[4] 王捷民，等. 基于改进的自适应遗传算法 HCGA 的测试数据自动生成[J]. 北京理工大学学报，2007，27(10)：883-885

[5] Mansour N, Salame-Jaffa M. Data Generation for Path Testing [J]. Software Quality Journal, 2004, 12: 121-136

[6] Miller J, Reformat M, Zhang H. Automatic test data generation using genetic algorithm and program dependence graphs [J]. Information and Software Technology, 2006, 48: 586-605

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

[8] Wegener J, Baresel A, Sthamer H. Evolutionary test environment for automatic structural testing[J]. Information and Software Technology, 2001, 43(14): 841-854

[9] 陈继锋，朱利，沈钧毅，等. 一种基于分支覆盖的测试数据自动生成算法[J]. 计算机科学，2006，33(12)：261-264

[10] 金虎，李志蜀，张磊，等. 基于面向路径的遗传算法的测试用例自动生成[J]. 计算机工程，2007，33(3)：21-23

[11] 巩敦卫，张岩. 一种新的多路进覆盖测试数据进化生成方法[J]. 电子学报，2010，38(6)：15-16

[12] 白凯，崔冬华. 基于遗传算法的测试数据自动生成[J]. 计算机与数字工程，2010，38(2)：52-54