第32卷 第23期 计 算 机 工 程 2006年12月
Vol.32 No.23 Computer Engineering December 2006
·软件技术与数据库· 文章编号：1000—3428(2006)23—0064—04 文献标识码：A 中图分类号：TP311

基于免疫遗传算法的构件化软件测试用例生成

马 臻[1]，张毅坤[1]，梁 荣[2]，鲁晓锋[1]，徐艳丽[1]，解建仓[3]
(1. 西安理工大学计算机科学与工程学院，西安 710048；2. 西安科技大学计算机系，西安 710054；
3. 西安理工大学水利水电学院，西安 710048)

摘 要：提出了一种基于免疫遗传算法(GA)的构件化软件测试用例生成模型(MTCGCBS)，介绍了IGA算法的基本思想。通过将IGA算法与传统遗传算法和随机算法在水利构件化软件测试用例生成中的比较，说明了IGA算法的效率明显高于传统遗传算法和随机算法，同时也进一步验证了模型的正确性、可行性。

关键词：免疫遗传算法；构件化软件测试用例生成模型；疫苗

Test Case Generation for Component-based Software Based on Immune Genetic Algorithm

MA Zhen[1], ZHANG Yikun[1], LIANG Rong[2], LU Xiaofeng[1], XU Yanli[1], XIE Jiancang[3]
(1. School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048;
2. Department of Computer, Xi'an University of Science and Technology, Xi'an 710054;
3. Institute of Water Resources and Hydre-electric Engineering, Xi'an University of Technology, Xi'an 710048)

【Abstract】 This paper suggests a model of test case generation of component-based software (MTCGCBS) based on an immune genetic algorithm (IGA), and introduces the basic thought of IGA. Through comparing IGA with both traditional genetic algorithm and random algorithm, IGA is further proved to be superior in generating test case for component-based software, and at the same time, the correctness of MTCGCBS is validated.

【Key words】 Immune genetic algorithm (IGA); Model of test case generation of component-based software(MTCGCBS); Vaccine

1 概述

随着软件规模的不断扩大，高效率和高质量的软件开发成为现代软件工程学研究的热点。基于构件的软件工程(Component-Based Software Engineering, CBSE)是为了保证高效、优质地进行软件开发而应运而生的技术。但是对于构件化软件来说，如何保证软件可靠性也是摆在测试工程师面前的难题，而软件测试正是保证软件可靠性的必要手段。因此基于构件化软件的测试技术必将受到测试工程师的重视。

在构件化软件测试技术中，高效的测试用例生成是简化测试工作、提高测试效率的必要手段。由于最初生成的测试用例数量庞大、测试效率低下，因此需要利用一种强有力的优化算法对最初生成的测试用例进行优化。遗传算法是一种多点搜索和采用交叉操作的技术，具有良好的全局搜索能力，但对于局部空间的搜索问题不是很有效，个体的多样性减少得很快。为了克服以上缺点，Chun等基于体细胞理论和免疫网络理论提出了一种免疫遗传算法[1]。在这个算法中，将抗原作为目标函数、抗体作为解答、抗原和抗体之间的亲和力作为解答的联合强度，显示了独特的优势和高效性。通过将其思想应用于构件化软件测试用例生成中就可以对测试用例进行优化从而大大简化软件的测试工作。

本文提出了一种基于免疫遗传算法(Immune Genetic Algorithm, IGA)的构件化软件测试用例生成模型(Model of Test Case Generation of Component-based Software, MTCGCBS)，并详细地介绍了IGA算法的基本思想，将其与传统遗传算法和随机算法在水利构件化软件测试用例生成中进行比较，进一步说明了IGA算法在构件化软件测试用例生成中的优越性，同时也验证了MTCGCBS模型的正确性、可行性。

2 基于IGA的测试用例生成模型

在构件化软件中，每个构件都有其规约文档。对根构件规约文档进行分析得到根构件的前置条件，然后利用随机算法生成构件化软件测试的初始测试数据，对初始测试数据进行编码得到初始种群，再利用IGA算法对初始种群进行优化，这样就可以产生高效率的测试用例。

图1所示的模型就是根据构件规约文档生成初始测试数据进而利用IGA算法进行优化的构件化软件测试用例生成模型。从图中可以清楚地看出如何根据构件规约文档得到初始测试用例集，还可以看到IGA算法是如何对初始测试用例集进行优化进而得到优化后的测试用例集的。

初步得到的测试用例数量是非常庞大的，可以利用IGA算法来对这些测试用例进行优化，在图1中可以看出IGA算法利用适应度函数和被测软件进行交互，每一个测试用例运行后都会返回给IGA算法一个适应度函数值，此值越高说明该测试用例的程序覆盖率越高，测试效率越高。这里的程序覆盖率是指构件化软件每个构件接口调用的方法被覆盖的比率。

基金项目：国家“863”计划基金资助项目(2005AA113150)
作者简介：马 臻(1981－)，男，硕士生，主研方向：软件测试；张毅坤，教授；梁 荣、鲁晓锋，助教；徐艳丽，硕士生；解建仓，教授、博导
收稿日期：2005-12-02 **E-mail：**mazhenapril@126.com

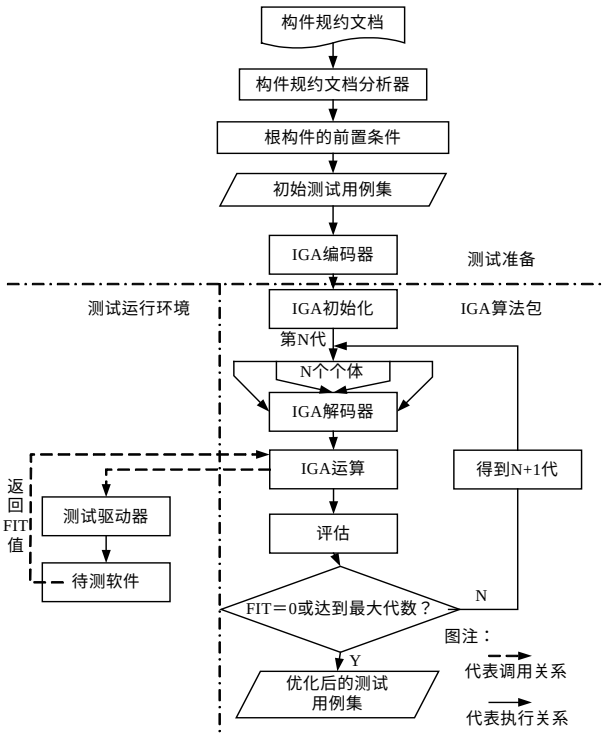

图 1 MTCGCBS 模型

3 IGA 算法

遗传算法是一类基于“ 产生+测试 ”方式的迭代搜索算法，尽管算法在一定条件具有全局收敛特性，但算法的选择、交叉、变异等操作一般都是在概率意义下随机进行的，虽然保证了种群的群体进化性，但一定程度上不可避免地有退化现象的出现。大量研究表明，仅仅依赖于以遗传算法为代表的进化算法在模拟人类智能化处理事务能力方面远远不足，还需要更深入地挖掘和利用人类的智能资源，而免疫遗传算法就是将生命科学中免疫的原理与遗传算法相结合来提高算法的整体性能，并有选择、有目的地利用待求解问题中的一些特征信息来抑制优化过程中退化现象的出现。

3.1 IGA 算法定义

免疫遗传算法是在传统遗传算法的基础上加入一个免疫算子，加入免疫算子的目的是防止种群退化。而免疫算子是由接种疫苗和免疫选择两个过程组成的。本文采用笔者先前研究的一种改进的遗传算法作为基本算法，再加入免疫算子从而构成本文的免疫遗传算法。

本文的免疫遗传算法(IGA)的形式化定义如下：

定义 1　IGA 算法可以定义为一个 11 元组：

$IGA=(C,F,P_0,M,\Phi,\Gamma,\Psi,\Omega,\Lambda,\Pi,T)$

式中，C：个体编码方法；

F：个体的适应度评价函数；

P_0 ：初始种群；

M：群体大小；

Φ ：选择算子；

Γ ：自适应交叉算子；

Ψ ：自适应变异算子；

Ω ：迁移概率P_S；

Λ ：迁移数量N_S；

Π ：免疫算子；

T：算法终止条件。

根据以上 IGA 算法定义可以给出 IGA 算法的流程，如图 2 所示。

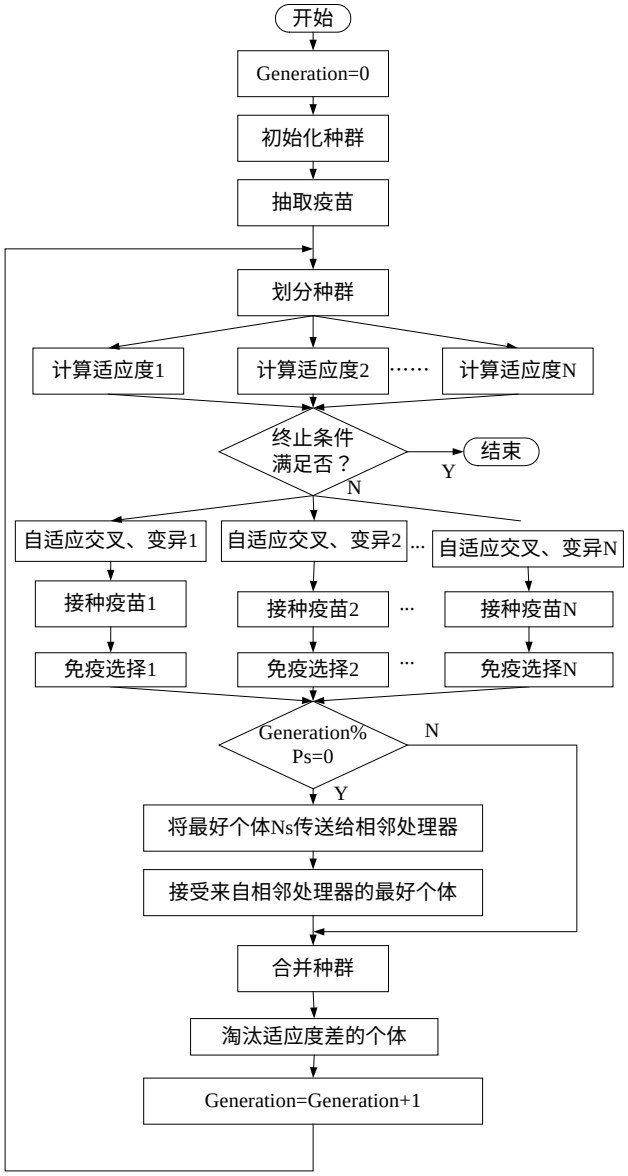

图 2 IGA 算法流程

3.2 IGA 算法收敛性

设种群初始规模为N，种群中所有个体均采用动态变长的二进制编码，交叉操作选择Uniform Crossover方法，其中交叉点的选取是通过自适应的动态交叉率P_C获得的[2]，变异操作是对每个基因位以自适应的动态概率P_M相互独立地进行变异。算法的状态转移情况可用随机过程

$$A_k \xrightarrow{\text{自适应交叉}} B_k \xrightarrow{\text{自适应变异}} C_k \xrightarrow{\text{接种疫苗}} D_k \xrightarrow{\text{免疫选择}} A_{k+1}$$

来表示，其中A_k到D_k的状态转移构成一个马尔科夫链。限于篇幅，证明过程不再给出，具体证明过程参见文献[3]。

3.3 IGA 算法描述

3.3.1 抽取疫苗

疫苗是从对所求问题的先验知识中提炼出来的。它所包含的信息量及其正确性对算法的性能起着重要的作用。

抽取疫苗的过程如下：

第 1 步：分析待求问题，搜集特征信息。

对于要解决的生成高效测试用例来说，就是在最短的时

间内用尽可能少的测试用例覆盖所有用到的构件方法调用路径，如果想生成最小的测试用例集，那么每个测试用例调用了哪些构件方法必须明确，这样就可以挑选那些执行不同构件方法调用路径的测试用例组成最小测试用例集。因此在构造适应度函数时就要对每个构件调用的方法做标记。

基于上述认识，只要在计算适应度函数时记录每个测试用例的构件方法调用路径就可以有针对性地选择所需要的测试用例，将那些执行了一定构件方法调用路径的测试用例加入测试用例集。所谓一定构件方法调用路径就是指测试用例集中的测试用例没有一个执行该路径。

第 2 步：根据特征信息制作免疫疫苗。

如果某一测试用例覆盖了较多的构件化软件的构件方法调用路径，那么在下一代中就可以将其优良的基因位保留，则可以定义免疫疫苗为：

如果满足：

$$v^i = \begin{cases} C^i, & \text{若 } C \text{ 的第 } i \text{ 位为优良基因位} \\ *, & \text{否则} \end{cases}$$

其中C为种群，v为C的疫苗模式(简称疫苗)，v^i为疫苗的第i位。

疫苗具有以下两个性质：

(1)种群的所有优良个体均具有该种群的疫苗。

(2)疫苗的平均适应度(适应度估计)不低于相应种群的平均适应度。

设种群C的个体为$c_1, c_2, \ldots, c_N$，C的疫苗抽取过程[4]描述如下：

(1)置模式 v 的所有基因位均为*；

(2)求出种群 C 的所有优良个体，记为 $y_1, y_2, \ldots, y_n (N \geq n)$；

(3) $i = 1$;

(4) $j = 1, x = y_j^i$;

(5)j=j+1，如果 j>n，转过程(7)

(6)如果 y_j^i =x，转过程(5)，否则转过程(8)

(7) $v^i = x$;

(8)i=i+1，如果 i>L，结束，否则转过程(4)。

其中 L 为编码长度。

3.3.2 适应度函数

适应度函数的构造成功与否直接影响到算法的成功与否。适应度函数的函数值表征了某个个体对应参数域中的测试用例的测试效率，适应度高的个体对应的测试用例测试效率就高，这样的测试用例可以在更短的时间内覆盖尽可能多的构件化软件路径，发现更多的构件化软件错误。在 IGA 算法中的每一代种群、每一个个体都要进行适应度计算，只不过鉴于 IGA 算法的并行性，这些运算都是由每个种群所在的单独的算法单元完成的，这样一来运算速度就可以得到明显提高，进而提高了 IGA 算法的效率。

对于 IGA 算法而言，并不需要了解所解决的具体问题是什么样的。具体问题解决的好坏都是由适应度函数构造的合适程度决定的。

构造适应度函数主要需要满足几个条件：

(1)单值、连续、非负、最大化；

(2)合理、一致性。要求适应度值反映对应解的优劣程度；

(3)计算量小。适应度函数设计应尽可能简单。这样可以减少计算时间和空间上的复杂度，降低计算成本；

(4)通用性强。适应度对某类具体问题，应尽可能通用，最好无需使用者改变适应度函数中的参数。

一般而言，适应度函数是由目标函数变换而成的。而对于构件化软件测试而言，用测试用例覆盖准则来表征测试用例的测试效率。覆盖率高的测试用例其测试效率就越高，进而其适应度函数的值就越高。

在构造适应度函数前首先要在构件化软件的每个构件前插桩一个标记函数，此标记函数可以唯一地标识相应的构件接口调用，假设某构件化软件共由 m 个构件组成，每个构件有 n 个接口，则每个构件前需插桩的标记函数可定义为

$$K_1 = f_1(x_1, x_2, \ldots, x_n)$$
$$K_2 = f_2(x_1, x_2, \ldots, x_n)$$
$$\cdots\cdots$$
$$K_m = f_m(x_1, x_2, \ldots, x_n)$$

而在软件结束处插入适应度函数

$$F = U(K_1) + U(K_2) + \ldots + U(K_m)$$

其中，$U(x) = \begin{cases} x, & x > 0 \\ 0, & x < 0 \text{ or } x = 0 \end{cases}$

3.3.3 自适应交叉、变异

交叉操作分为 2 个步骤：第 1 步将选择产生的个体随机地两两组合；第 2 步根据交叉率P_C进行交叉操作。

目前许多学者认为交叉率应该随着遗传进程而自适应地变化，这样有组织的进化具有更高的鲁棒性、全局最优性和效率[5]。因此本文构造交叉率为

$$P_C = \begin{cases} P_{C1} - \dfrac{(P_{C1} - P_{C2})(F' - F_{avg})}{F_{\max} - F_{avg}}, & F' \geq F_{avg} \\ P_{C1}, & F' < F_{avg} \end{cases}$$

其中，$F_{\max}$是种群中最大的适应度函数值；F_{avg}是每代群体的平均适应度函数值；F'是要交叉的 2 个个体中较大的适应度函数值；P_{C1}, P_{C2}是初始化时第 1、第 2 代的交叉概率。

尽管交叉操作对于 IGA 算法而言很重要，但不能保证不会遗漏一些重要的遗传信息。在 IGA 算法中，变异算子用来防止这种不可弥补的遗漏。变异就是某个个体的某一基因偶然地随机改变。当它有节制地和交叉算子一起使用时，它就成为一种防过渡成熟而丢失重要信息的保证策略。

变异率P_M越大，种群的多样性就越好，发生早熟的可能性就越小，但较大的变异率P_M将使IGA算法退化为随机搜索，进化速度变慢。同样，太小的变异率P_M又不能达到变异操作的效果。故本文构造变异率为

$$P_M = \begin{cases} P_{M1} - \dfrac{(P_{M1} - P_{M2})(F_{\max} - F)}{F_{\max} - F_{avg}}, & F \geq F_{avg} \\ P_{M1}, & F < F_{avg} \end{cases}$$

其中，F 是要变异个体的适应度函数值；P_{M1}, P_{M2}是初始化时第 1、第 2 代的变异概率。

3.3.4 接种疫苗

设个体 x，给其接种疫苗是指按照先验知识来修改 x 的某些基因位上的基因或其分量，使所得个体以较大的概率具有更高的适应度。这一过程应该满足如下 2 点：

(1)若个体 y 每一基因位上的信息都是错误的，即每一位码都与最佳个体不同。则对任意个体 x，x 转为 y 的概率为 0；

(2)若个体 x 每个基因位都是正确的，即 x 已是最佳个体，则 x 以概率 1 转为 x。

设有群体$C = (x_1, x_2, \ldots, x_N)$，对 C 接种疫苗是指在 C 中按比例$\alpha$随机抽取$n_\alpha = \alpha N$个个体而进行的操作。疫苗是从

对问题的先验知识中提炼出来的，它所包含的信息量及其准确性对算法的性能起着重要的作用。

3.3.5 免疫选择

这一过程分为 2 部分：(1)免疫监测，即对接种了疫苗的个体进行监测。若其适应度不如其父代，说明在自适应的交叉、变异的过程中出现了严重退化现象。这时，该个体将被父代中所对应的个体取代；(2)利用蒙特卡罗法来进行选择，即在目前子代中以概率：

$$P_i = \frac{F_i}{\sum_{i=1}^{m} F_i}, \qquad i = 1, 2, \ldots, m$$

选择个体进入新的父代种群。其中 m 为种群数，F_i 为第 i 个个体的适应度函数值。

4 模型、算法验证

MTCGCBS 模型和 IGA 算法的验证是通过对“面向水利信息化的应用集成中间件平台及其应用”课题中的构件化软件部分进行测试用例生成来完成的。生成的测试用例目标是用最少数量的测试用例来覆盖 100%的构件方法调用。图 3 是 3 种算法在生成测试用例数量上的比较，图 4 是 3 种算法在运行时间上的比较。

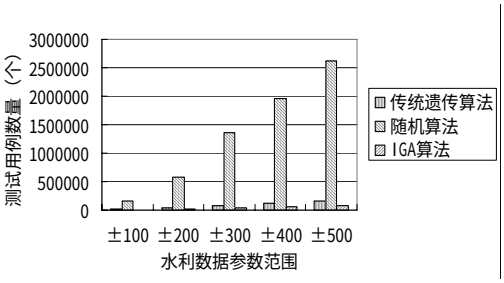

图 3　3 种算法测试用例数量比较

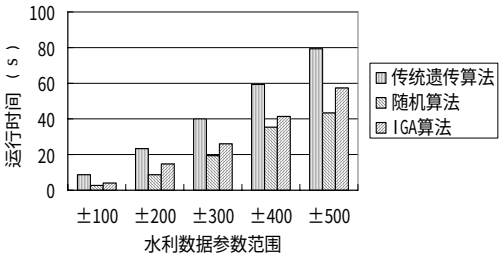

图 4　3 种算法运行时间比较

从图中可以看到，IGA 算法比传统遗传算法和随机算法所需的测试用例数量明显要少，也就是说，IGA 算法可以用很少的测试用例就可以达到 100%的构件方法调用覆盖率，这正是 IGA 算法所追求的目标。这里 IGA 算法比传统遗传算法运行时间短，是因为 IGA 算法引入了并行性思想；IGA 算法比随机算法所运行的时间较长，是因为 IGA 算法的复杂程度造成的。相比运行时间而言，较少的测试用例是更有实用意义的。

5 结束语

本文提出了一种基于 IGA 算法的构件化软件测试用例生成模型，并将此模型应用于水利构件化软件的测试用例生成中，不仅说明了 IGA 算法在构件化软件测试用例生成中的效率明显高于传统遗传算法和随机算法，同时也进一步验证了模型的正确性、可行性，为构件化软件测试自动化打下了坚实的基础。

参考文献

1 王　凌. 智能优化算法及其应用[M]. 北京: 清华大学出版社, 2003.

2 Sthamer H H. The Automatic Generation of Software Test Data using Genetic Algorithms[D]. University of Glamorgan Prifvsgol Morgannwg, 1995-11.

3 王　磊, 焦李成. 免疫进化计算理论及应用[D]. 西安: 西安电子科技大学, 2001-09.

4 韩学东, 洪炳熔, 孟　伟. 基于疫苗自动获取与更新的免疫遗传算法[J]. 计算机研究与发展, 2005, 42(5): 740-745.

5 王小平, 曹立明. 遗传算法－理论、应用与软件实现[M]. 西安: 西安交通大学出版社, 2004.

6 Srinivas M, Patnaik L M. Adaptive Probabilities of Crossover and Mutation in GA[J]. IEEE Trans. on Systems, Man and Cybernetics, 1994, 24(4): 656-667.

7 Beydeda S, Gruhn V. An Integrated Testing Technique for Component-based Software (accepted)[C]. Proc. of AICCSA ACS/IEEE International Conference on Computer Systems and Applications, Beirut, Lebanon, 2001-06.

8 Wu Ye, Chen Meihwa, Offutt J. UML-based Integration Testing for Component-based Software[C]. Proceedings of the 2nd International Conference on COTS-based Software Systems, Ottawa, Canada, 2003-02.

9 Jorgensen P C. 韩　柯译. 软件测试[M]. 北京: 机械工业出版社, 2003-02.

10 聂长海, 徐宝文. 一种最小测试用例集生成方法[J]. 计算机学报, 2003, 26(12).

（上接第 48 页）

过程中，利用从时域 Petri 网模型中获得的数据，通过流程中是否存在死锁、制造提前期、流程执行时间减少率和资源成本等情况的分析，得到业务流程重组的评估方法，达到对流程充分了解和进一步的优化的目的，分析结果将成为业务流程重组方案实施的重要依据。

参考文献

1 陈禹六, 李　清. 经营过程重构与系统集成[M]. 北京: 清华大学出版社, 2001-06.

2 李人厚. 基于高级 Petri 网的协同设计中协调建模与应用的研究[D]. 西安: 西安交通大学, 2003.

3 乔　非, 高桂花, 白海涛. 基于高级 Petri 网的流程建模支持系统设计、开发与应用[J]. 计算机工程, 2000, 26(10).

4 Jiang Z B, Zuo M J, Fung Y K, et al．Temporized Colored Petri Nets with Changeable Structure(CPN-CS) for Performance Modeling of Dynamic Production Systems[J]. International Journal of Production Research, 2000, 38(8).

5 Chen H. Control Synthesis of Petri Nets Based on S-decreases[J]. Discrete Event Dynamic Systems: Theory and Application, 2000, 10(3): 233-249.

6 Liu C M, Wu F C. Using Petri Nets to Solve FMS Problems[J]. International Journal of Computer Integrated Manufacturing, 1993, 6 (3): 175-185.