基于免疫算法的 HLR 软件系统测试用例生成*)

马 臻 陈 天 李 元 孙 苗 郭 锐
(中国联通陕西分公司 西安 710075)

摘 要 提出了一种基于免疫遗传算法(Immune Genetic Algorithm,简称 IGA)的 HLR 软件测试用例生成模型(Model of Test Case Generation of HLR Software,简称 MTCGHS),并详细地介绍了 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 HLR 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,同时进一步验证了模型的正确性、可行性。

关键词 免疫遗传算法,HLR 软件测试用例生成模型,疫苗

Method of Generating Test Case for HLR Software Based on Immune Genetic Algorithm

MA Zhen CHEN Tian LI Yuan SUN Miao GUO Rui
(China Unicom Shanxi Branch, Xi'an 710075, China)

Abstract This paper suggested a model of test case generation of HLR software (MTCGHS) based on an immune genetic algorithm (IGA), and introduced the basic thinking of IGA in details. Through Comparison of IGA with both Traditional Genetic Algorithm and random Algorithm, IGA is further proved to be superior in generating test case for HLR software, and at the same time, the correctness of MTCGHS is validated.

Keywords IGA, MTCGHS, Vaccine

1 引言

随着 HLR 在通信系统中的作用越来越重要,功能越来越庞杂,HLR 软件系统的稳定性也越来越受到通信软件工程师的重视。HLR 系统软件可靠性如何保证,也成为软件测试工程师面临的难题。

在 HLR 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 HLR 软件测试用例生成中就可以对测试用例进行优化,从而大大简化软件测试工作。

本文提出了一种基于免疫遗传算法(Immune Genetic Algorithm,简称 IGA)的 HLR 软件测试用例生成模型(Model of Test Case Generation of HLR Software,简称 MTCGHS),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 HLR 软件测试用例生成中进行比较,进一步说明了 IGA 算法在 HLR 软件测试用例生成中的优越性,同时验证了 MTCGHS 模型的正确性、可行性。

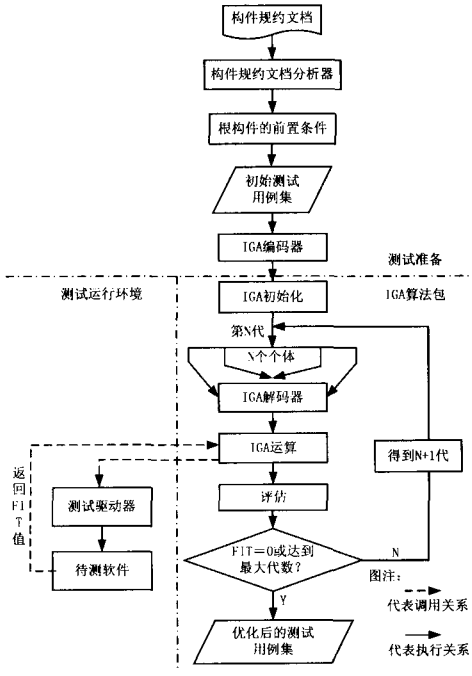

图 1 MTCGHS 模型

2 基于 IGA 的测试用例生成模型(MTCGHS)

在 HLR 软件系统中,每个功能都由一个独立的功能模块组成,每个模块都有其规约文档。对根模块规约文档进行分析,得到根模块的前置条件,然后利用随机算法生成 HLR 软件测试的初始测试数据。对初始测试数据进行编码,得到

*)基金项目:国家 863 高技术研究发展计划项目(2005AA113150)支持。马 臻 硕士,通信工程师,主要研究方向为通信软件测试、移动通信网络优化、电子渠道管理。

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模块规约文档生成初始测试数据，进而利用 IGA 算法进行优化的 HLR 软件测试用例生成模型。从图中我们可以清楚地看出如何根据模块规约文档得到初始测试用例集的，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，我们可以利用 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 HLR 软件每个模块接口调用的方法被覆盖的比率。

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出现退化现象。大量研究表明，仅仅依赖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：

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

其中，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 算法的流程图(图 2)。

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。限于篇幅所限，证明过程不再给出，具体证明过程参见文献[3]。

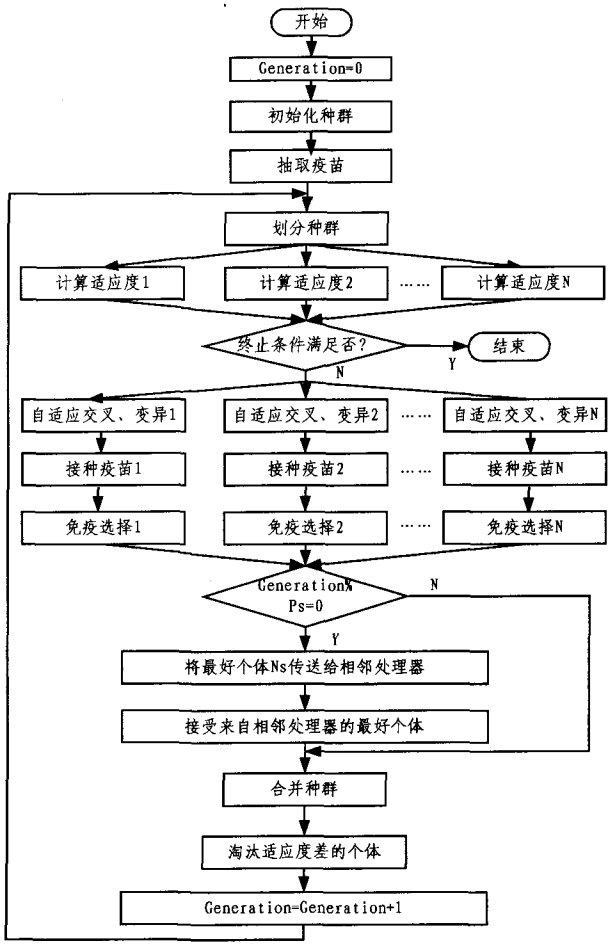

图 2 IGA 算法流程图

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：

第一步，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模块方法调用路径。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模块方法必须明确，这样就可以挑选那些执行不同模块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，就要对每个模块调用的方法做标记。

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模块方法调用路径，就可以有针对性地选择所需要的测试用例，将那些执行了一定模块方法调用路径的测试用例加入测试用例集。所谓一定模块方法调用路径，是指测试用例集中的测试用例没有一个执行该路径。

第二步，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软件的模块方法调用路径，那么在下一代中就可以将其优良的基因位保留，免疫疫苗可以定义为：

如果满足：

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

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：

性质 1 种群的所有优良个体均具有该种群的疫苗。

性质 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 $c_1, c_2, \cdots, c_N$，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 $y_1, y_2, \cdots, y_n (N \geqslant n)$；

(3)$i=1$；

(4)$j=1, x=y_j^i$；

(5)$j=j+1$，如果 $j>n$，转(7)；

(6)如果 $y_j^i=x$，转(5)，否则转(8)；

(7)$v^i=x$；

(8)$i=i+1$，如果 $i>L$，结束，否则转(4)。其中 L 为编码长度。

3.3.2 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两点：

(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 $C=(x_1, x_2, \cdots, x_N)$，对 C 接种疫苗是指在 C 中按比例 α 随机抽取 $n_a=\alpha N$ 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。

4 模型、算法验证

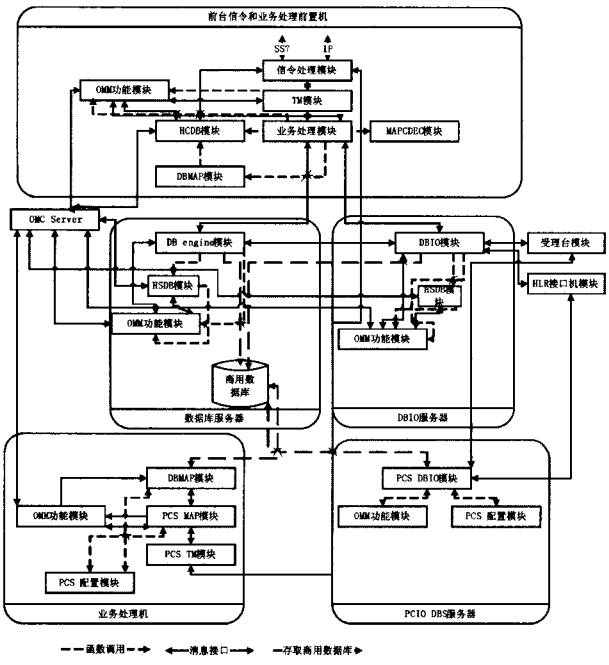

图 3 HLR 系统模块结构图

MTCGHS 模型和 IGA 算法的验证是通过对 HLR 软件进行测试用例生成来完成的。HLR 软件的结构图如图 3 所示。生成的测试用例目标是用最少数量的测试用例来覆盖 100%的模块方法调用。图 4 是三种算法在生成测试用例数量上的比较，图 5 是三种算法在运行时间上的比较。

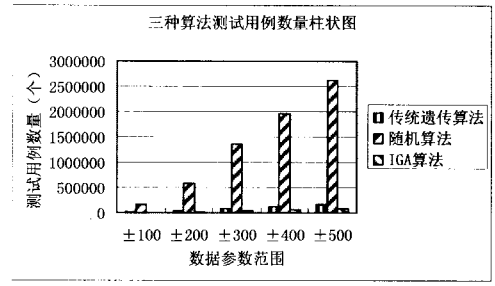

图 4 三种算法测试用例数量比较

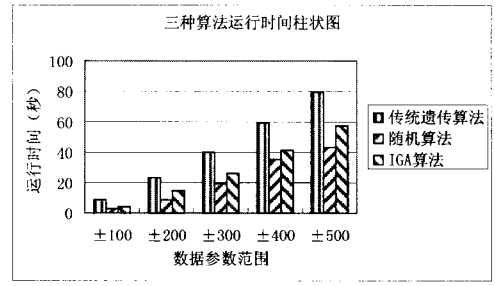

图 5 三种算法运行时间比较

从图中我们可以看到，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%的模块方法调用覆盖率，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 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。

结束语 本文提出了一种基于 IGA 算法的 HLR 软件测试用例生成模型，并将此模型应用于 HLR 软件的测试用例生成中，不仅说明了 IGA 算法在模块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，为模块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]. 清华大学出版社，2003

[2] Sthamer H-H. The Automatic Generation of Software Test Data Using Genetic Algorithms. A thesis submitted in partial fulfilment of the requirements of the University of Glamorgan / Prifvsgol Morgannwg for the degree of a Doctor of Philosophy, November 1995

[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

[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. 遗传算法一理论、应用与软件实现[M]. 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 compo-

(下转第 272 页)

• 生成输出模型树：根据输入模型树的遍历结果，判断输入模型树的节点与目标输出模型树的节点之间的映射关系，生成输出树的内容。

上述步骤中的源和目标节点之间的映射关系主要是根据源和目标模型的描述要求和特定来确定，一般是在元模型间进行映射。

5.3.3 模型转换结果

依据上述分析，在实际生成的过程中主要做好对图 6 中 GME 模型树的搜索和生成工作。在生成过程中所依赖的映射关系是 4.2 节中对图 3 元模型的描述和解释。转换后的 StateFlow 图如图 9 所示。只要 Matlab 环境中做一定的设置就可以进行数控车床工作模式的仿真，仿真结果将直接作为系统设计的依据。

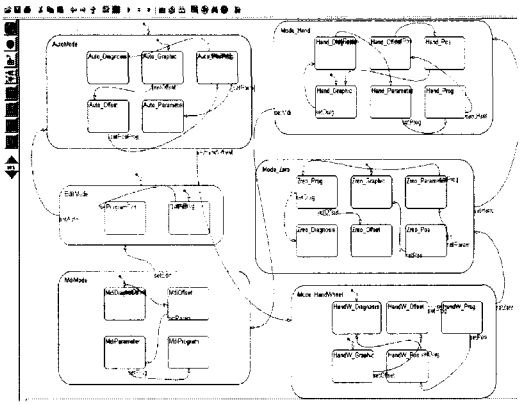

图 9 图 6 对应的转换后的 StateFlow 图形

结束语 本文总结了嵌入式数控系统开发方法的各种不足之处，经综合分析提出采用特定领域模型驱动开发方法来克服传统开发手段中的种种弊端。在此基础上提出嵌入式数控系统模型层验证框架，该框架突出了模型验证及领域建模语言在基于模型驱动开发中的重要地位。并通过基于 StateFlow 数控系统逻辑功能验证实例，描述了在该框架下系统建模语言的构建、模型转换技术的具体实现手段。特定领域模型驱动开发为嵌入式数控系统的开发提升了相应的抽象层次，改进了现有开发质量和设计效率，从方法理论上为嵌入式数控系统提供创新。但目前该方法还有很多值得改进的地方，比如数控系统的各种非功能特性如何用图形化元建模语言来表示、具有特定语义数控领域平台的模型库的建设。另外，目前实现的模型转换是采用文本的方式直接转换，是否可以采用基于图文法的方式，从而提高模型转换的可操作性，这些将是下一步研究的重点内容。

参 考 文 献

[1] 国家中长期科学和技术发展规划纲要. http://www.gov.cn/jrzg/2006-02/09/content_183787.htm

[2] 国务院关于加快振兴装备制造业的若干意见. http://www.gov.cn/jrzg/2006-06/29/content_322286.htm

[3] 广东省工业产业结构调整实施方案(修订版)

[4] Rastofer U, Bellosa F. Component-based software engineering for distributed embedded real-time systems // Software, IEEE Proceedings-[see also Software Engineering, IEEE Proceedings. 2001, 148(3): 99-103

[5] Karsai G, Gray J. Component generation technology for semantic tool integration // Aerospace Conference Proceedings. 2000 IEEE, 2000, 4: 491-499

[6] 林湁，等. 基于实时组件的可重构数控系统研究. 小型微型计算机系统，2004，25(7)：1151-1154

[7] 魏仁选，周祖德，陈幼平，等. 可重用面向对象数控软件及其开发环境研究. 华中理工大学学报，1999，2(3)：19-21

[8] Wang Shige, Shin Kang G. Architecture for Embedded Software Integration with Reusable Components

[9] Wang Shige, Shin Kang G. Constructing Reconfigurable Software for Machine Control Systems. IEEE Transactions on Robotics and Automation, 2002, 18(4)

[10] MDA Guide. Version 1.1.1. Document Number: OMG/2112-16-11. OMG

[11] 高军礼. 基于模型驱动开发方法的开放式结构计算机数控系统的研究[D]. 博士学位论文. 华南理工大学

[12] 杜道山，李从心. 模型驱动在数控系统开发中的应用研究[J]. 青岛大学学报(工程技术版)，2005，20 (3)：53-59

[13] Selic B, Leo M. Using Models in Real-time Software Design [J]. IEEE Control Systems Magazine, 2003, 23(3): 31-42

[14] Martin G, Lavagno L, Louis-Guerin J. Embedded UML: a merger of real-time UML and co-design[C] // Proceedings of the 9th International Symposium on Hardware/software Codesign (CODES01). Copenhagen, 2001: 23-28

[15] Domain-Specific Modeling, 10 times faster than UML. http://www.metacase.com/[EB/OL]

[16] 郑人杰，殷人昆，陶永雷. 实用软件工程[M]. 清华大学出版社，1997

[17] 程国达，彭澄廉. 嵌入式系统描述与验证环境的实现[J]. 计算机辅助设计与图形学学报，2004，16(1)：109-115

[18] UPPAAL. http://www.uppaal.com/[EB/OL]

[19] Sztipanovits J, Karsai G. Model-integrated Computing[J]. IEEE Computer, 1997, 30(4): 110-112

[20] Sendall S, et al. Model transformation: The heart and soul of Model-driven Software Development. IEEE Software, 2003: 42-45

[21] Karsai G, Gray J. Component Generation Technology for Semantic Tool Integration[C]. IEEE Aerospace Conference Proceedings, 2000, 4: 491-499

(上接第 246 页)

nent-based software (accepted) // AICCSA ACS/IEEE International Conference on Computer Systems and Applications. Beirut, Lebanon, June 2001

[8] Wu Ye, Chen Mei-Hwa, Offutt J. UML-based Integration Testing for Component-based Software // Proceedings of 2nd International Conference on COTS-Based Software Systems. Ottawa, Canada, Feb. 2003

[9] Jorgensen PC. 软件测试[M]. 韩柯，等译. 北京：机械工业出版社，2003

[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)：12

[11] 景旭，黄东. 移动通信 HLR 软件子系统的设计与研究[D]. 南京：东南大学，2006