ISSN:2661-3719(Print); 2661-3727(Online)

基于机器学习的编译器测试优化方法研究

邵语辰
上海市宝山区上海大学宝山校区　201900

摘　要：在系统软件中，编译器占据着十分重要的地位。如果编译器自身存在问题，就会对可执行文件造成严重的影响，导致文件出现错误。在编译器质量管理中，通过相应的测试手段进行检测十分必要。在近年来的技术发展中，对编译器进行自动测试的技术得到了发展。但当下的测试中大多对测试用例生成的工具有较大的依赖性，例如Csmith等。在进行测试时通过生成大量的测试用例实现功能测试。除此之外，编译器自身也是一个十分庞大的软件，因此在测试中相关的执行人员也会对其进行压力测试，通过使用大量的测试用例来实现这一过程，但这一测试存在较为严重的效率问题。

关键词：机器学习；编译器测试；方法优化

在如今的计算机领域中，编译器在软件开发中存在着至关重要的影响。受此影响，编译器测试工作逐渐成为计算机研究领域中的热门话题。在软件开发中，软件测试环节是必不可少的一个步骤，也是保证软件质量的关键，通过编译器能帮助设计人员发现并纠正软件中存在的缺陷。但在实践当中，测试中存在着oracle问题，在对编译器进行测试中，相关的研究人员通过采用随机差异、差异以及蜕变测试技术，在一定程度上解决了oracle问题在测试中带来的肤面影响，对研究人员测试结果分析起到了很大的促进作用，并帮助研究人员针对问题进行追溯定位。但在现有的技术当中，对编译器进行测试的实践周期很长，并且通过测试客户发掘的错误量也非常低，在测试中由于编译器自身较大，由此带来了较为严重的效率问题。为了加快对编译器测试的质量和效率，对当前的测试方法进行优化十分必要。

一、编译器测试技术

编译器在软件工程中发挥着不可忽视的作用，为了保证其运行的质量，相关研究人员先后提出了大量的编译器测试技术。除此之外，还有一些研究人员对当前的测试技术进行了实验研究，对比不同编译器的测试性能。在此对当前主流的三种编译器测试技术进行简述[1]。

（一）随机差异测试

该项测试技术在编译器测试中的应用十分广泛。在具体使用中，是通过多个编译器之间进行相互比较，以此实现对其功能揭错的效果。如果测试输入条件相同的情况下，没有错误发生时，其输出的结果会是相同的。但如果输出结果出现差异，就可以判定使用的编译器中存在问题，出现问题的编译器可能是一种，也可能是几种。当在随机差异测试中使用两个以上的编译器参与时，在测试结果出现差异时，可以通过投票的方式寻找出错误的编译器，其基本流程如图1（a）所示。其中C1，C2，…，Cn代表可比较编译器的数量，其中O1，O2，…，On代表输入条件下P和P的输入I的输出。

（二）差异优化级别测试

这种测试方式是随机差异测试的一种变体，两者之间的差异是差异优化测试能实现在相同输入的前提下，采用不同优化级别参数来编译输入程序，在测试中如果得到的可执行文件结果相同，则证明编译器无问题。否则则说明该编译器存在问题。其基本的方法流程如图1（b）。在此之中L_1，L_2，…，L_n代表编译器C的编译优化级别，O_1，O_2，…，O_n代表输入条件下P和P的输入I的输出。

（三）等价模输入方法

这也是一种行之有效的编译器检测办法。在进行测试中，首先根据现有程序生成等价程序，基于此作为编译器测试的输入用例。在完成测试后，通过与原始测试相互对比的方式，作为输入用例测试结果的对照。通过与原始测试数据得到的测试结果与生成的等价测试结果进行对比，基于此判断编译器是否工作正常。其整体的流程图如图1（c）所示。其中V_1，V_2，…，V_n代表原始

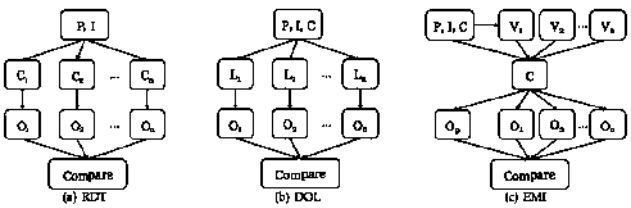

图1　当前三种主流编译器流程图

测试程序输入下等价变体，O_1，O_2，…，O_n代表测试输入的输出，O_p表代表测试输入程序P情况时的输出。

二、编译器测试加速

在上述的测试技术中，大多十分依赖测试用例自动生成工具，通过工具的方式生成测试程序完成对编译器的测试工作。但这种测试办法存在十分严重的效率问题，在当前这一问题十分明显。在近年来的对编译器优化方法的研究中，一些针对测试效率提升的办法被相继提出，并逐渐加以应用。在此之中使用较为广泛的是TB-G和LET方法。其中TB-G检测办法是对测试文本程序进行TB-G。在具体使用中，将测试程序看作是文本，在检测中采用抽取的方式提取可能出现错误的片段，并最终将其转变为文本向量。在此之中，与错误有关的字符包括操作字符、语句字符以及类型字符等。在获得相应的文本向量后，通过TB-G方法将这些内容标准化加工。其主要的流程如图2所示。

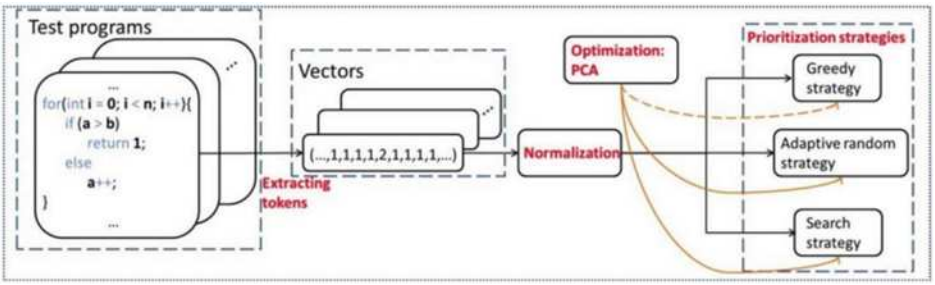

图2　TB-G基本流程图

相比之下，LET方法对编译器测试进行加速的效果更好一些。在该检测方法中，包含了学习调度的过程。在学习阶段中，该方法对错误程序的相关特点进行定义，在此之后通过这些特征对数据进行预处理，在此之后生成训练模型数据。在LET训练中，会对每一个程序进行揭错评估，预测期揭错的可能性，同时对程序执行时间进行计算预测。在调度阶段中，该检测方法会通过相应的模型，对每一个测试程序的揭错可能性以及执行时间进行预测，并对结果进行计算，在最后根据这两个维度对测试程序进行合降序排序，以此确定最终的程序测定序列[2]。

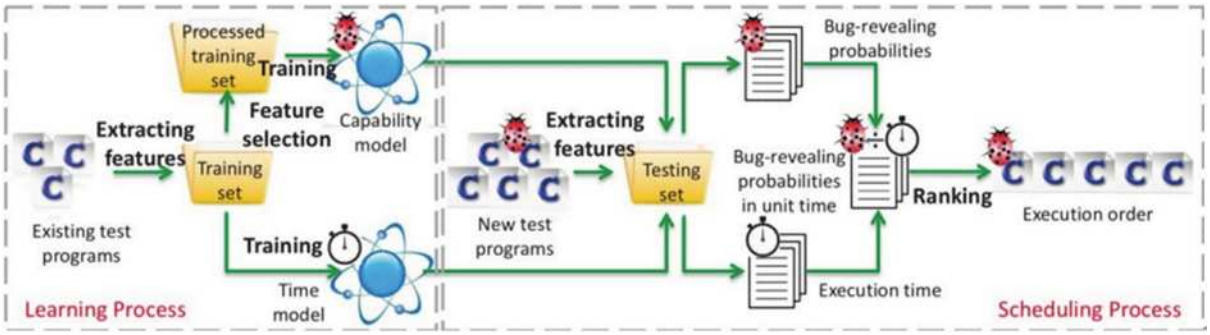

图3　LET基本流程图

在传统的测序方法使用中，存在较为明显的效率低下的问题，难以实现提高检测效率的目标。但在LET这种检测方式中，该检测方法能实现自适应排序的效果。并且在对测试用例进行选择中，会选择其中差异最大的用例，在对差异性进行衡量中，一般通过距离来实现。在相关测试统计中发现，自适应随机排序的办法在实践使用中也存在效率低的问题，在具体使用中该办法在排序中花费了大量的代价。

三、编译器测试加速方法

在软件工程当中，编译器是非常重要的基础软件。在对编译器质量的管理中，测试技术尤为关键。但当前的测试技术中，大多需要通过大量的测试用例方式作为输入，以此实现对编译器的测试效果。但这种方法势必会消耗大量的实践，存在严重的效率低下问题。在近年来对此的相关研究中，为了进一步优化编译器的测试能力，提高其测试的效率，相关研究人员提出了优先执行的办法，即对最有可能揭错的程序进行测试，因此实现对编译器测试效率提高的效果。但这些测试方式在实践中却存在一个严重的问题，在不同的测试程序在编译中，有可能出现触发编译器相同错误的可能，这回严重影响到对编译器测试的效率提升。在此通过使用覆盖率进行测试，能有效的实现对重复揭错的测试程序进行区分的

效果。但是对覆盖信息的搜集中，对资源消耗十分巨大。因为覆盖信息生成过程是动态的。相关研究人员基于覆盖信息的基础上，对测试程序进行聚类区分，以此实现提高效率的效果，并将其命名为COP，在此分为三个部分对该方式进行探讨[3]。

（一）覆盖信息的预测

在编码器对一个测试程序进行变异种，在这一过程中产生的足尖信息会被覆盖。例如函数是否被执行等信息。这就是编译中的覆盖信息。在对其进行预测中，可以将这一过程视为多输出的回归问题，通过标签的方式，代表对组件中被关注的各种元素，例如代码是否被执行等。这种方式与传统的机械学习算法的应用方式十分相似，在对覆盖信息进行预测中，需要对首先进行特征定义、制作标签、预测等。

（二）预测程序的聚类

在成功获取程序测试中的覆盖信息后，COP会对这些信息进行整理，将其归类为不同的种类，这些不同种类的测试程序在测试中，会有很大可能触发不同的错误。该方法就是通过聚类算法实现聚类，选择X-means聚类算法的目的是该方式你不需要对待分类的数量提前确认。

（三）测试程序的排序

在基于测试程序分类结果的基础上，COP能实现对测试程序的分类。类别的不同往往会导致待测试编译器出现不同的错误。在此之中，COP会首先通过LET的方式对待测程序所需要消耗的时间以及揭错概率进行统计计算，然后对程序进行选择，在每一类中选出单位时间中揭错概率最大的程序进行测试，并根据该结果对测序循序进行排序。并且在测试中会选定测试阈值，当单位时间中揭错概率大的用例被选择完毕后，才会进一步选择单位时间概率小的用例作为测试程序。

四、结束语

综上所示，在当前对编译器进行测试中，存在较为严重的效率问题。在此背景下很多测试方法相继被提出。但这些测试方法主要是基于一些准则的方式，对程序用例进行重新排序的方式，通过优先执行的办法选出出发错误可能最高的程序进行测试。这种方法虽然能在一定程度上实现提高速度的效果，但却忽视了在不同测试程序中，也有可能发生相同错误的问题。这种情况极大程度上现有测试方法产生了肤面的影响。在程序测试当中，其覆盖信息的差异越大，这些程序处罚不同错误的可能性就越大。但当前的程序测试过程都是在动态环境下生成的，因此对信息搜集带来了很大的困难。通过COP的方法对覆盖信息的有效性进行甄别，这对提高整体的测试效率有非常大的帮助效果。

参考文献：

[1]曾伟明，薛云志，赵琛，贺也平.一种编译优化测试用例自动生成方法的设计与实现[J].小型微型计算机系统，2009，30（01）：13-18.

[2]饶彦，陈伟.基于编译器控制流结构测试用例的生成[J].计算机时代，2005（04）：34-35.

[3]蒋立源，黄广君.一个编译器的测试用例自动生成系统[J].西北工业大学学报，1992（02）：153-158.