软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2022,33(9):3297−3311 [doi: 10.13328/j.cnki.jos.006387]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

软件测试路径选择优化模型及其进化求解*

杜 莹[1], 孙百才[1], 巩敦卫[1], 田 甜[2], 姚香娟[3]

[1](中国矿业大学 信息与控制工程学院, 江苏 徐州 221116)

[2](山东建筑大学 计算机科学与技术学院, 山东 济南 251100)

[3](中国矿业大学 数学学院, 江苏 徐州 221116)

通信作者: 巩敦卫, E-mail: dwgong@vip.163.com; 田甜, E-mail: tian_tiantian@126.com

摘 要: 路径测试是一种非常重要且应用广泛的结构测试方法, 已有路径生成方法的测试效率不高、测试开销较大, 且易生成冗余测试路径. 针对以上问题, 主要研究路径选择问题的优化模型及其进化求解方法, 目的在于: 在不降低测试覆盖率的前提下, 减少冗余路径的数量, 降低测试消耗. 首先, 以多条路径作为决策变量, 基于该决策变量包含的边数和路径数, 建立多目标优化模型; 然后, 采用多目标进化算法求解该模型, 得到目标路径集. 将所提方法应用于 7 个基准测试程序, 并与其他算法比较. 实验结果表明, 相比其他算法, 所提方法能够在保证测试充分性的条件下, 降低测试消耗, 从而提高测试效率.

关键词: 路径测试; 路径选择; 多目标优化; 带精英策略的非支配排序遗传算法; Pareto 最优解集

中图法分类号: TP311

中文引用格式: 杜莹, 孙百才, 巩敦卫, 田甜, 姚香娟. 软件测试路径选择优化模型及其进化求解. 软件学报, 2022, 33(9): 3297–3311. http://www.jos.org.cn/1000-9825/6387.htm

英文引用格式: Du Y, Sun BC, Gong DW, Tian T, Yao XJ. Optimization Model of Path Selection for Software Testing and Its Evolution-based Solution. Ruan Jian Xue Bao/Journal of Software, 2022, 33(9): 3297–3311 (in Chinese). http://www.jos.org.cn/1000-9825/6387.htm

Optimization Model of Path Selection for Software Testing and Its Evolution-based Solution

DU Ying[1], SUN Bai-Cai[1], GONG Dun-Wei[1], TIAN Tian[2], YAO Xiang-Juan[3]

[1](School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China)

[2](School of Computer Science and Technology, Shandong Jianzhu University, Jinan 251100, China)

[3](School of Mathematics, China University of Mining and Technology, Xuzhou 221116, China)

Abstract: Path testing is a very important and widely used structural testing method. Existing path generation methods are either time-consuming or labor-intensive, or they can generate a large number of redundant paths. To solve the above problem, this work mainly studies the optimization model of path selection problem and its evolutionary solution method. The purpose is to reduce the number of redundant paths and reduce test consumption without reducing test coverage. First, a number of paths are selected as the decision variable, and the number of edges and paths included in these paths are taken as the objective to formulate a multi-objective optimization model. Then, the multi-objective evolutionary algorithm is employed to solve the formulated model with the purpose of obtaining the target path set. The proposed method is applied to test 7 benchmark programs and it is compared with the existing method and greedy algorithm. Experimental results show that, compared with other algorithms, the proposed method can reduce the test consumption under the condition of ensuring test sufficiency, thereby improving the test efficiency.

Key words: path testing; path selection; multi-objective optimization; non-dominated sorting genetic algorithm with elitism; Pareto-optimal solution set

* 基金项目: 国家自然科学基金 (61773384, 61763026, 61673404); 国家重点研发计划 (2018YFB1003802-01); 山东省自然科学基金 (ZR2020MF084)

收稿时间: 2021-01-10; 修改时间: 2021-03-02, 2021-05-17; 采用时间: 2021-05-25; jos 在线出版时间: 2022-01-28

1 引 言

软件测试是软件质量评价和改进的过程, 也是软件质量保证的重要手段[1], 贯穿整个软件开发生命周期. 然而, 软件测试的工作量很大, 据统计, 测试时间会占到总开发时间的 40%, 一些可靠性要求非常高的软件, 测试时间甚至占到开发周期的 60%[2]. 因此, 寻找降低软件测试耗时的方法日益迫切.

程序的路径由控制流图上执行控制流经过的节点和有向边组成, 路径数量随程序分支的增加呈指数增长, 因此, 对所有路径进行覆盖是非常困难的. 此外, 程序可能存在不可达路径, 为不可达路径生成测试数据所需的消耗不但很大, 而且没有意义. 路径选择通常面向目标覆盖的元素, 通过合适的选择策略能够减少目标路径的数量, 并选出可达路径. 最少路径选择策略和最少谓词策略是两种常见的路径选择策略[3], 其中, 最少路径选择策略利用最少的路径覆盖所有的元素; 最少谓词策略选择路径时优先考虑覆盖判定节点最少的路径.

本文提出的是一种采取最少路径选择策略的方法, 该方法选择能保证测试充分性的最小路径子集, 以实现和初始路径集相同的边覆盖率. 主要思想为: 首先, 以若干路径组成的路径集作为决策变量, 以覆盖的边数和决策变量中路径的条数为目标, 建立多目标优化模型; 然后, 采用基于 NSGA-II 的进化求解方法求解该模型, 得到需要测试的路径集.

本文的贡献主要体现在: (1) 建立了路径选择问题的多目标优化模型; (2) 提出了一种基于 NSGA-II 的模型求解方法; (3) 验证了多目标进化算法应用于求解路径选择问题的有效性.

本文第 2 节综述相关工作; 第 3 节是本文的重点, 阐述提出的方法, 包括: 路径选择问题优化模型和基于 NSGA-II 的进化求解方法; 第 4 节是所提方法在基准程序测试的应用, 以及与其他方法的对比实验结果和分析; 最后, 第 5 节总结全文, 并指出进一步需要研究的问题.

2 相关工作

本节从如下两个方面, 总结已有的研究工作, 包括: 结构测试的相关概念和路径集的相关生成方法, 从而引出本文的研究动机.

2.1 结构测试

结构测试的方法总体上分为静态分析和动态分析两大类. 路径测试是一种重要的动态分析技术, 该方法通过设计足够的测试数据覆盖程序中所有可能的路径. 路径编码是路径测试数据生成的基础, 已有的编码方式很多, 有的采用程序的语句编号[4]序列表示, 有的采用分支节点序列[5]表示. 但是, 这两种方式不仅路径表示序列很长, 而且在表示不同路径时会产生冗余编码. 鉴于此, 我们曾提出路径的 Huffman 编码表示[6], 进一步提高路径测试效率. 但是此方法面对逻辑结构很复杂的程序时, 匹配度的计算是不合理的. 为解决该问题, Wei 等人将边界符号加入 Huffman 编码中[7]. 此改进方法可以有效地分离每个分支, 每个迭代和每个循环. 此外, 丁蕊等人提出了一种关键点路径表示法[8], 通过该表示法可将路径分成易覆盖、难覆盖和不可行路径, 更快的生成测试用例. 对于含有多个进程的并行程序, Sun 等人[9]将多个进程的路径组合成一条路径进行编码.

对于覆盖测试数据生成, 目前已有许多研究成果, 主要有 4 种方法, 即随机法、静态法、动态法和启发式方法. 其中, 随机法通过对输入空间随机采样得到测试数据; 静态法通过对程序静态分析得到测试数据; 动态法通过执行程序得到测试数据; 启发式方法使用诸如遗传算法及其改进方法[10−14]的元启发式优化技术自动生成测试数据.

2.2 路径集生成

实际上, 即使一个不太复杂的程序, 包含的路径数目往往也很多, 这使得完整的路径测试变得非常困难[15]. 为了解决该难题, 只能把覆盖的路径压缩到一定范围内. 基路径测试就是这样的一种测试方法. 在 20 世纪 80 年代, McCabe 提出基路径的概念, 将程序中所有可达的路径集合看成一个向量空间, 对应向量空间中的基, 存在一组基路径, 对基路径的线性组合能够覆盖整个路径集合, 因此, McCabe 认为找出路径空间中的一组基路径进行测试, 如果这组基路径没有问题, 那么, 用这组基路径表示的一切路径组合都没有问题[16].

基本路径测试法是白盒测试的测试方法中运用最为广泛的一种, 基本路径测试法是在程序控制流图的基础上, 通过分析控制构造的环路复杂性, 导出基本可执行路径集合, 从而设计测试用例的方法. 设计出的测试用例要保证在测试中程序的每个可执行语句至少执行一次. 基本路径集是程序的部分路径集合, 它满足下面 3 个性质: 每条路径都至少包含一条其他路径未包含的边, 也就是每一条都是独立路径; 程序中所有的边都被基路径集中的路径所覆盖; 程序中的所有的、不属于基路径集的路径都能由基路径集中的路径通过线性运算获得.

对于基路径集的生成, McCabe [17]提出一种基线路径法, 该方法通过被测程序的控制流图寻找基路径集, 是一种比较有效的测试方法. 但是, 当程序的逻辑结构复杂时, 此方法的适用性不强, 且难以计算圈复杂度, 从而降低了基路径测试的覆盖率和效率[18]. 针对基路径测试中路径不能自动生成的问题, 国内外学者进行了相关研究, Wijayasiriwardhane 等人[19]将程序流程图转化为有向图, 并自动计算有向图的环数. 王冠等人[20]对基路径测试中 Cabe 方法进行了改进, 提出通过数组生成基路径的方法, , 提高了基路径生成的正确性. Zolotov 等人[18]提出一种高速结构测试模型, 为基路径测试中路径相关性的度量提供了依据, 解决了无效路径的问题. 上述方法主要是对 McCabe 方法的改进, 有些无法包含所有的边, 有些很难在基路径测试中应用, 且适用的程序结构有限[21].

目前, 代表性的基路径生成方法有如下几种. Ghiduk [21]利用个体长度可变的遗传算法生成基路径, 夏良等人[22]基于遗传算法对选取的个体适应值调整, 达到保留必须存在的测试路径, 减少重复测试路径的目的. Srivastava 等人[23]使用某些规则集, 通过蚁群优化算法找出所有有效路径. Antonia 等人[24]通过简化的控制流图生成路径, 实现了对程序中所有语句、分支的覆盖. 杜庆峰等人[25]在简化控制流图的同时, 使得生成的基路径集含有的不可达路径尽可能的少. 此外, 还有许多其他路径生成方法, 如以数据流测试的 ALL-DU-PATHS 准则作为标准选取测试路径集合[26]等. 上述方法由于将源代码直接转化为有向图和程序路径, 忽略了代码中的逻辑信息, 得到的基路径集在程序运行中往往不可达, 导致无法生成测试数据. Zhang 等人[27]提出基于分支相关性的基路径集生成方法, 减少了不可达路径, 并最大限度的包含了相关的分支. 韩寒[15]在考虑路径可达性的基础上, 得到了包含尽可能多可达路径的基路径集. 该方法的独立可达路径数小于圈复杂度, 但依然以圈复杂度为标准选取路径. 以上方法都无法得到包含所有可达边的最小路径集, 得到的路径集中都含有大量冗余路径, 即路径集中的某些子集也能满足测试需求, 冗余路径指的就是路径集中除去子集后剩余的那些没有必要再测试的路径.

由于本文是在采用文献 [21] 的方法得到基路径集中选择更小的满足条件的路径集, 在这里简要介绍一下该方法. 该方法分为以下 4 个模块.

(1) 分析模块: 在该模块中, 通过阅读被测程序, 得到控制流图、DD 图和后继表并传递到下一模块.

(2) 路径生成模块: 首先, 通过被测程序的 DD 图的入口和出口边缘初始化所有路径. 然后, 通过选择、交叉、变异和育种 (添加新边) 操作最终获得完整的路径. 最后, 将生成的路径传递给下一模块.

(3) 可行性检查模块: 调用一个约束求解器对生成的路径进行可行性检查, 如果路径可行, 则将路径传递给下一模块, 否则将路径传递给路径生成模块生成另一条路径.

(4) 独立性检查模块: 如果生成的路径是独立路径, 则算法将该路径添加到基本路径集中. 否则, 算法会忽略这条路径并开始新的循环来寻找不同的完整路径.

该方法每次生成一个测试路径. 因此, 该方法将重复此过程, 直到生成一组基本测试路径或满足停止条件.

通过以上步骤我们可以看出, 每次生成一个可行的完整独立路径都会将其放入基本路径集中, 这样虽然保证了 100% 的边覆盖率, 但是在基本路径集中很可能有某些子集同样能实现 100% 的覆盖率, 除去子集之外的那些路径就是冗余路径.

在基于模型的测试方面, Chen 等人[28]分解扩展有限状态机, 转化为扩展有限状态树, 并自动计算通过树的测试状态转换路径. 冯利航[29]采用基于迁移覆盖的集合进化算法生成测试序列.

2.3 路径选择

王思岚等人[30]提出了单元覆盖测试中基于区间运算的路径选择, 该方法首先选定一段从函数入口到目标覆盖元素的半条路径, 然后根据区间运算判定该路径的可达性并改造不可达路径, 进而得到一条包含目标覆盖元素的可达路径. 李青翠等人[31]提出一种修正条件判定覆盖测试的路径选取方法. 该方法介绍了一种基于抽象语法树

提取原子谓词和独立影响对的方法, 提出了一种基于程序控制流图选择待测真值向量和路径的方法. 为了解决循环路径的问题, Gao 等人[32]提出了一种新的路径探索方法, 减少了实现高覆盖率所需的状态数, 该方法先通过确定状态优先级, 然后修剪不改变代码覆盖率状态而创建的新状态. 吴川等人[33]研究了回归测试的分支覆盖问题, 通过利用已有测试数据的路径覆盖信息, 并选择一定个数的路径, 以覆盖所有的目标分支. 冯金金[34]提出了基于不同函数之间的关系确定权值的办法, 通过建立加权复杂网络, 发现软件系统的关键节点, 得到关键测试路径, 进而对软件测试路径进行聚类分析, 压缩测试路径. Prema 等人[35]提出一种利用节点树选择简化测试路径集的算法, 该方法首先从节点树中找到测试路径集, 然后根据测试路径的权重对测试路径进行划分, 并从每一个分区中选择一个测试路径.

可以看出, 关于基路径生成和路径选择的研究并不多, 可以将其大致分为两种, 一种是为了覆盖全部节点而不断选取路径, 此类方法易生成冗余测试路径, 测试开销较大; 另一种是先对路径进行分类, 然后在各自的类中选取代表路径, 此种方法可能无法覆盖所有节点, 测试效率不高. 鉴于这些已有的方法存在很多问题, 都不利于提高测试效率, 本文针对已有方法生成的基路径集中, 含有大量冗余路径的问题, 首次考虑将路径选择问题建模为多目标优化问题, 原因是多目标优化问题能够同时衡量多个目标, 然后通过求解建立的模型, 得到需要覆盖的路径集. 这样一来, 目标路径集在包含控制流图中所有可达边的同时, 不含冗余的路径, 从而提升测试效率.

3 提出的方法

本节建立路径选择问题的数学模型并给出其进化求解方法. 首先, 以多条路径作为决策变量, 基于该决策变量包含的边数和路径数, 建立多目标优化模型; 然后, 采用多目标进化算法求解该模型, 得到 Pareto 最优解集, 基于该解集, 形成目标路径集.

可以看出, 多目标优化模型建立和基于 NSGA-II 的模型求解, 是本文需要解决的关键技术. 在第 3.1 节至第 3.2 节, 将详细阐述这些关键技术.

3.1 路径选择问题模型的建立

通过已有方法能够得到一个基路径集合, 作为本文的初始路径集 X'. 该集合中的路径不仅能包含控制流图中的所有可达边, 而且都是可达的路径. 然而, 该方法得到的初始路径集合中有大量的冗余路径, 这些多余的路径将大大增加测试消耗. 本节和第 3.2 节将采用提出的方法解决该问题, 得到不含有冗余路径的目标路径集.

本文中, 去掉冗余路径的问题实质上是一个路径选择问题, 可以描述为: 从被测程序中选择一定数量的路径形成目标路径集, 使得该路径集既不含有冗余路径, 基于该路径集的测试又能够满足测试充分性准则[36]. 鉴于此, 本节从初始路径集中, 选择若干路径形成目标路径集, 使得该路径集包含的可达边尽可能多、路径尽可能少. 可以看出, 路径选择问题能够建模为一个多目标优化问题. 下面, 建立该问题的多目标优化模型.

(1) 决策变量

在传统的优化问题中, 决策变量一般为单个元素, 而本文中路径选择问题的决策变量为若干路径形成的集合, 记为 $X=\{x_1,x_2,\ldots,x_m\}$, 其中, $x_i(i=1,2,\ldots,m)$ 为该集合的第 i 条路径, TM 为该集合包含的路径条数. 由于每一路径均由若干条边组成, 因此, 路径 x_i 可表示为 $x_i=\left(e_1^i,e_2^i,\ldots,e_{n_i}^i\right)$, 其中, $e_j^i(j=1,2,\ldots,n_i)$ 为路径 x_i 包含的第 j 条边, n_i 为该路径包含边的条数. 记初始路径集 X' 的幂集为 $2^{X'}$, 由于目标路径集 X 是一个子集合, 因此, $X\in 2^{X'}$, $2^{X'}$ 为优化问题的搜索空间.

(2) 目标函数

首先, 我们希望选择的路径集包含的边最多. 这是因为, 包含的边越多, 那么, 选择的路径集越能够满足测试充分性准测. 记决策变量对应的路径集 X 包含边的集合 $E=x_1\cup x_2\cup\ldots\cup x_m$, 那么, 该路径集包含的边数可以表示为:

$$f_1(X)=|E|=|x_1\cup x_2\cup\ldots\cup x_m| \tag{1}$$

然后, 我们希望选择的路径集不包含冗余路径, 那么, 路径数就要尽可能的少. 这是因为, 路径越少, 需要生成的测试数据就越少, 从而降低路径覆盖测试的时间消耗. 记 $|X|$ 为 X 包含的路径条数, 那么,

$$f_2(X) = -|X| \tag{2}$$

这样一来, 建立的路径选择问题的多目标优化模型可以表示为:

$$\begin{cases} \max f(X) = (f_1(X), f_2(X)) \\ \text{s.t. } X \in 2^{X'} \end{cases} \tag{3}$$

公式 (3) 的目标是期望 f_1 和 f_2 都能最大, 一般来说, 如果某路径集包含的边数越多, 那么, 需要的路径也应该越多, 但我们却期望该路径集包含的路径少. 因此, 公式 (3) 包含两个相互冲突的目标, 是一个典型的两目标最大化问题. 此时, 需要采用有针对性的方法, 高效求取问题的 Pareto 最优解集.

3.2 基于多目标进化算法的模型求解

多目标算法有多种, 多目标粒子群和多目标遗传算法是较常用的两种算法, 由于本文研究的是离散型问题, 考虑到 PSO 算法的应用大多数注重连续优化问题, 在离散优化问题上研究较少. 因此, 鉴于 NSGA-II 是目前解决多目标优化问题最广泛、效果最好的方法之一, 本文采用 NSGA-II 求取问题的 Pareto 最优解集.

此外, 还需根据问题的特性, 确定进化种群合适的初始化方法和产生新解的方式. 这些是将 NSGA-II 方法求解模型时, 关键的步骤, 其他关于 NSGA-II 的详细内容, 请参阅文献 [37].

(1) 种群初始化

对于公式 (3) 表示的优化模型, 决策变量 X 由若干分量组成, 每一分量代表一条路径, 每条路径可用整数表示, 因此, 本文对种群的个体采用整数编码. 考虑到本文的一个优化目标是路径集包含的路径条数, 也即个体规模, 因此, 个体规模是一个需要变化的量, 变化区间为 $[1,|X'|]$. 在种群进化过程中, 为了使个体规模能够通过交叉操作改变, 需要在种群初始化时, 保证种群中个体的规模不同.

种群初始化方法如算法 1 所示. 首先, 让种群 *pop* 和个体 $X_l(l=1,2,\ldots,P)$ 为空, 并复制初始路径集 X', 得到副本 X_c'; 然后, 生成 P 个 1 到 $|X'|$ 之间的随机整数, 作为种群中每一个体的规模; 接着, 对 X' 进行不放回抽样, 将抽到的路径放入 X_l. 此时, 判断 X' 是否为空集. 如果是, 那么, 对 X_c' 进行放回抽样; 否则, 继续对 X' 进行不放回抽样. 重复以上过程, 直到形成的个体 X_l 满足规模要求, 并将该个体加入种群中; 最后, 记种群规模为 P, 重复以上步骤 P 次, 得到初始种群 *pop*.

为了便于理解, 不妨假设种群有 3 个个体, 这些个体包含的路径数分别为 3、5 和 4, 个体编码如图 1 所示.

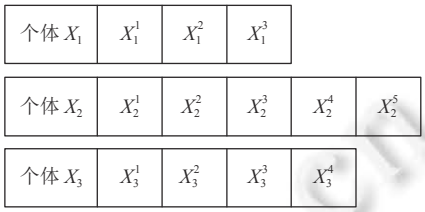

图 1 种群中 3 个体的编码

算法 1. 种群初始化.

输入: 初始路径集 X';

输出: 初始种群 *pop*.

BEGIN

1. 初始化种群 *pop* 为空;
2. 复制初始路径集 X', 得到副本 X_c';
3. FOR $l=1 \rightarrow popsize$ DO
4. 随机生成一个 1 到 $|X'|$ 之间的整数作为 $X'size$;
5. 初始化 X_l;
6. i=1;
7. WHILE $|X_l|$!=X_l *size* DO
8. WHILE X'不是空集 DO
9. 从 X'中随机选择 X_l';

10. 将 X_l^i 放入 X_l 并将其从 X' 中删除;
11. $i \leftarrow i+1$;
12. END WHILE
13. 从 X_c' 中随机选择 X_l^i;
14. 将 X_l^i 放入 X_l;
15. $i \leftarrow i+1$;
16. END WHILE
17. 将 X_l 加入种群 *pop*;
18. END FOR
19. RETURN *pop*;

END

(2) 个体交叉与变异

在寻找目标路径集的过程中, 通过交叉操作能够改变个体规模, 产生新的个体. 由上节可知, 本文的个体是一个集合, 既可以通过两个集合之间的交叉操作产生新个体, 又可以通过集合内部的交叉操作产生新个体. 由于集合内部的交叉操作会使新产生的个体包含不可达或不完整路径, 因此, 本文采用集合之间的交叉操作. 在交叉过程中, 首先, 对所有个体配对; 然后, 根据交叉概率判断配对的两个体是否需要执行交叉操作. 如果需要, 无论当前两个体的规模是否相同, 都分别在两个体中随机选择一个交叉点; 最后, 将两个体从各自交叉点到末尾的基因互换. 这样一来, 能够保证种群中个体规模的多样性, 有助于找到问题的 Pareto 最优解集.

记待交叉的两个体为 $X_1=(x_1^1,x_1^2,\ldots,x_1^\alpha)$ 和 $X_2=(x_2^1,x_2^2,\ldots,x_2^\beta)$, α 和 β 分别为 X_1 和 X_2 的个体规模. 记 x_1^u 为 X_1 的交叉点, x_2^v 为 X_2 的交叉点, 交叉之后产生的新个体分别为 $X_{1'}$ 和 $X_{2'}$, 那么, 交叉过程如图 2 所示.

对于一个集合个体来说, 可以有多种变异操作, 如增减基因、更改基因长度、更改基因内部. 本文选择的变异操作为通过更改基因内部, 产生新个体. 为保证变异过程中路径的完整性和可达性, 本文采用整条路径直接替换的方式. 首先, 依据个体变异概率判断需要执行变异操作的个体: 然后, 随机选择一个需要变异的基因位, 即变异点; 最后, 从初始路径集中随机选择一条路径, 判断该路径是否与当前个体中的路径重复. 如果不重复, 那么, 采用该路径替换变异点上的路径; 否则, 重新选择一条路径. 如果初始路径集中不存在这样的路径, 那么, 不执行替换操作.

记需要变异的个体为 $X_1=(x_1^1,x_1^2,\ldots,x_1^\alpha)$, 选择的变异点为 x_1^ω, 用来替换的路径为 $x_1^{\omega'}$, 变异之后的新个体为 $X_{1'}$, 那么, 变异过程如图 3 所示.

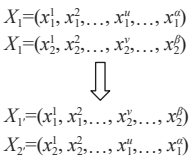

图 2 个体的交叉过程

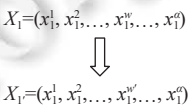

图 3 个体的变异过程

基于上面给出的相关函数和算法, 利用 NSGA-II 求解目标路径集的伪代码如算法 2 所示. 算法中, *popsize* 为种群规模, *MaxGen* 为最大进化代数.

算法 2. 求解目标路径集的 NSGA-II.

输入: 初始路径集 X';

输出: 目标路径集 X''.

BEGIN

1. 执行算法 1, 得到初始种群 *pop*;

2. FOR $j=1\rightarrow MaxGen$ DO
3. FOR $l=1\rightarrow popsize$ DO
4. 计算 f_1 和 f_2;
5. END FOR
6. 非支配排序, 计算拥挤距离;
7. 选择精英个体, 并将其保留;
8. 进行交叉、变异操作;
9. END FOR
10. 得到 Pareto 最优解集;
11. 选择包含边数最多的路径集作为目标路径集 X'';
END

4 实 验

本节通过一系列实验, 验证所提方法的有效性. 首先, 提出需要验证的问题; 然后, 介绍被测程序; 描述实验过程之后, 提供并分析实验结果.

4.1 需要验证的问题

为了评价所提方法的性能, 提出如下需要验证的问题.

问题 1. 所提方法选择的目标路径集, 能否包含控制流图中所有可达边?

为了回答该问题, 首先, 分别计算所提方法、MOPSO 算法以及文献 [35] 所提方法选择的路径集中包含的边数 E''和控制流图中所有可达边数 E'; 然后, 通过 $E''/E'\times 100\%$ 计算包含率. 如果包含率为 100%, 那么, 所提方法选择的目标路径集, 能够包含所有可达边.

问题 2. 所提方法选择的目标路径集, 能否大幅度减少路径数量?

为了回答该问题, 我们将所提方法选择的路径条数分别与文献 [21] 所提方法得到的初始路径集中路径条数、MOPSO 算法、贪心算法、整数规划方法[38,39]以及文献 [35] 所提方法选择的路径条数进行对比. 首先, 分别计算所提方法、MOPSO 算法、贪心算法、整数规划以及文献 [35] 所提方法选择的路径集中路径条数 $|X''|$和初始路径集中路径条数 $|X^*|$; 然后, 通过 $(|X^*|-|X''|)/|X^*|\times 100\%$ 分别计算约减率. 如果约减率很大, 那么, 所提方法选择的目标路径集, 能够大幅度减少路径数量.

问题 3. 所提方法采用多目标优化选择路径集, 能否提高测试数据生成效率?

为了回答该问题, 首先, 将所提方法选择的路径集作为待覆盖路径集, 计算覆盖该路径集的测试数据生成时间; 然后, 将文献 [21] 所提方法和选择问题中常用的贪心算法得到的路径集作为待覆盖路径集, 计算覆盖该路径集的测试数据生成时间. 如果所提方法选择的路径集生成测试数据所需的时间比其他两种方法短, 那么, 所提方法能够提高测试数据生成效率.

4.2 测试程序

选取 7 个不同规模和难度的基准和工业程序作为被测程序, 这些程序按照代码行数排序, 基本信息如表 1 所列, 表中, Triangle 和 Day 选自文献 [40]; Numday 选自文献 [41]; Totinfo、Replace、Space 和 Flex 选自西门子系统, 其源代码可以从网站 Software-artifact Infrastructure Repository 免费下载.

4.3 实验过程

实验平台的硬件配置如下: Inter Core i7-8750H CPU、8 GB 内存和 512 GB 固态硬盘; 软件配置为: Windows 10 操作系统和 Visual Studio 2013 编译器.

实验中, 首先, 采用已有方法[21], 得到未约简的基路径集, 即初始路径集, 基于此确定决策变量和目标函数; 然

后, 设置 NSGA-II 的控制参数, 考虑到文献 [42] 采用 NSGA-II 解决 EFSM 的可行测试序列生成问题与本文的问题类似, 参考该文献, 设定交叉概率为 0.7, 变异概率为 0.1; 对于 Triangle, 取种群规模为 20, 进化代数为 10; 对于 Day、Numday、Totinfo 和 Replace, 取种群规模为 100, 进化代数为 20; 对于 Space 和 Flex, 取种群规模为 500, 进化代数为 100; 最后, 采用文献 [43] 提出的传统遗传算法, 生成覆盖目标路径集中每一路径的测试数据, 此时, 遗传算法的控制参数设置如下: 个体采用二进制编码, 采用轮盘赌选择、单点交叉和单点变异, 且交叉和变异概率分别为 0.9 和 0.3; 对于 Triangle、Day 和 Numday, 取种群规模为 50, 进化代数为 2 000; 对于 Totinfo, 取种群规模为 70, 进化代数为 10 000; 对于 Replace, 取种群规模为 80, 进化代数为 10 000; 对于 Space 和 Flex, 取种群规模为 100, 进化代数为 30 000; 此外, 为了减轻随机因素对实验结果的影响, 所有实验独立进行 10 次, 并统计这些实验结果的平均值.

表 1 被测程序的基本信息

程序	代码行数	程序功能
Triangle	35	三角型分类
Day	42	计算某天的次序
Numday	248	计算两个日期之间的天数
Totinfo	406	计算统计信息
Replace	564	模式匹配和替换
Space	6 199	数组定义语言解释器
Flex	10 459	语法分析生成器

根据本文研究的问题, 本文设计了相应的多目标离散粒子群算法[44]求解该模型并将其作为对比方法. 更新种群中进化个体的位置与速度的过程如公式 (4).

$$\begin{cases} V^{t+1} = w \otimes V^t \oplus C_1 \otimes (P_{id} - X_{id}) \oplus C_2 \otimes (P_{gt} - X_{id}) \\ X^{t+1} = X^t \oplus V^{t+1} \end{cases} \tag{4}$$

其中, 速度保留值 w 为 2, 全局最优值和个体最优值对进化的影响因数 C_1 和 C_2 均为 4. 为了公平比较, 在每个测试程序中 MOPSO 与 NSGA-II 设置相同的种群规模和相同的最大迭代次数. 本文采用超体积 (HV) 策略[45]评估多目标算法的性能. 相对多目标优化中其他测度, 如 GD 和 IGD 测度, 该测度无需事先知道问题的真实最优解集; 更为重要的是, 该测度可以同时评估出结果的分布性和收敛性. 所得解集的分布性和 (或) 收敛性越好, HV 值就会越大. 考虑某一算法所得 Pareto 最优解集, 其 HV 测度值为这些最优解与参考点所围成区域的体积. 通常, 参考点由所得 Pareto 最优解集中关于每个目标函数的最差结果组成. 此外, 选择多目标优化中常用的 SC 测度[45]来比较两种算法的收敛性. 该测度主要用来计算集合 A 中占优元素 B 中的比例. 不妨设 A 和 B 是两个解集, SC 测度的计算公式如下:

$$SC(A,B) = \frac{|\{b \in B : \exists a \in A, a \succ b\}|}{|B|} \tag{5}$$

其中, $|\cdot|$表示集合中元素的数目. 显然, 当 $SC(A, B)$ =1 时, 集合 A 中元素全部占优集合 B; 而当 $SC(A, B)$= 0 时, 集合 A 中任何元素都不占优集合 B.

为了公平比较, 同所提方法和 MOPSO 一样, 整数规划也是自行实现的求解程序.

4.4 实验结果与分析

4.4.1 关于问题 1 和问题 2

为了回答问题 1 和问题 2, 首先, 使用所提方法得到 Pareto 最优解集; 然后, 选择包含最多可达边的路径集, 即 Pareto 前沿中最右侧点代表的路径集; 最后, 判断选择的路径集能否包含所有可达边, 并大幅减少路径数量.

图 4 是 7 个被测程序的 Pareto 前沿, 横坐标表示路径集中路径的条数, 纵坐标表示路径集包含边数的负值. “▷”和“+”表示本文所提基于 NSGA-II 的方法和基于 MOPSO 方法得到的 Pareto 前沿.

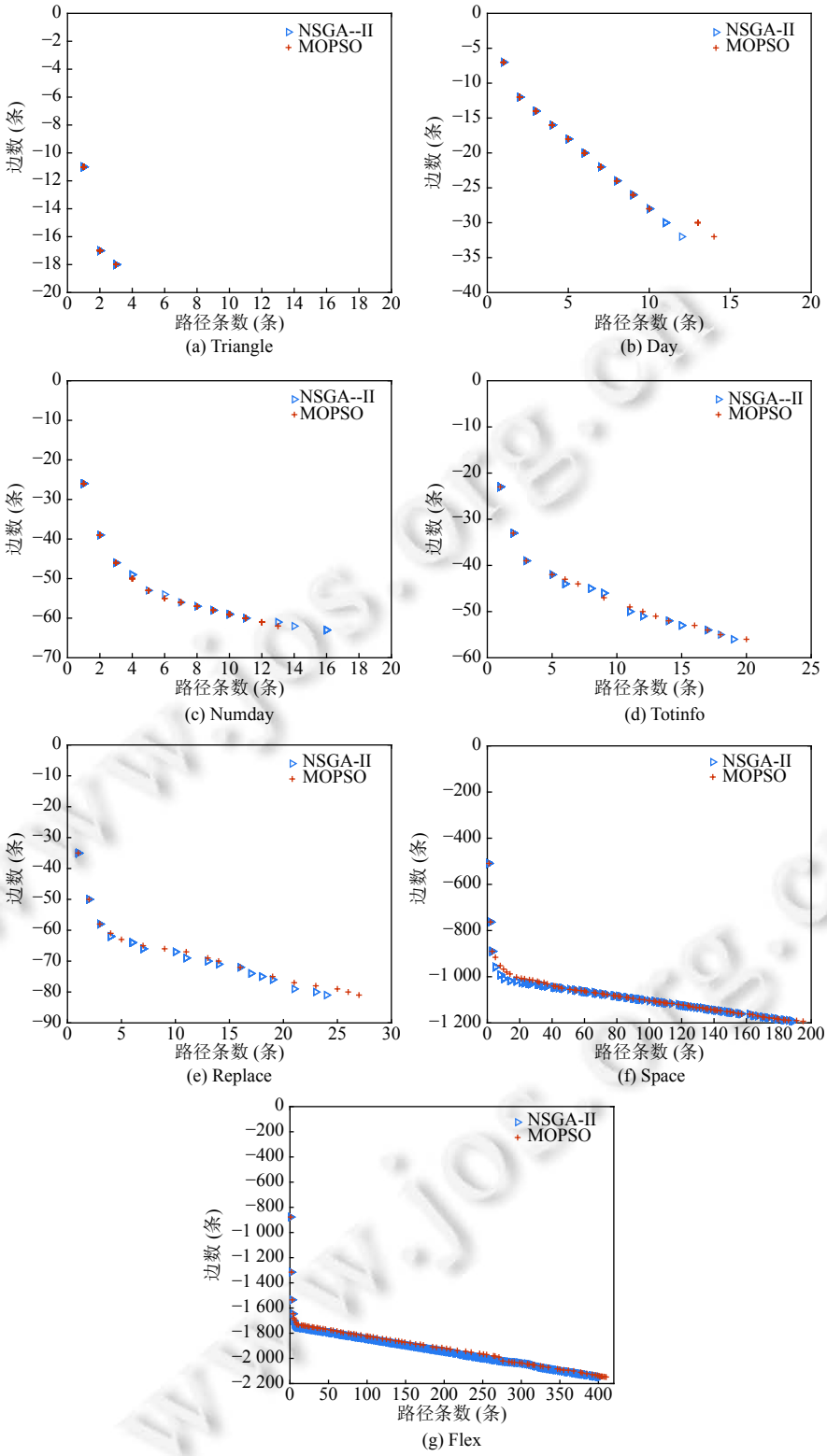

图 4　7 个程序的 Pareto 前沿

表 2–表 5 列出所选路径集包含的边数和包含率、路径条数和约减率, 表 2 中, 第 2–5 列分别为所提方法选择的路径集包含的边数、控制流图中所有可达边数、MOPSO 算法、贪心算法、整数规划算法及文献 [35] 方法选择路径集的边数; 表 3 中, 第 2–4 列分别为所提方法、基于 MOPSO、贪心算法、整数规划算法和文献 [35] 所提方法选择路径集的包含率. 且包含率越高, 包含的边数越多; 表 4 中, 第 2–7 列分别为所提方法、已有方法[21]、MOPSO、贪心算法、整数规划方法及文献 [35] 所提方法选择的路径集中路径条数和约减率; 表 5 中, 第 2–6 列分别为所提方法、MOPSO、贪心算法、整数规划方法及文献 [35] 所提方法选择的路径集的约减率; 表 6 列出不同多目标方法的 HV 值和 SC 值, 第 2–5 列分别为所提方法的平均 HV 测度、MOPSO 的平均 HV 测度、所提方法占优 MOPSO 的 SC 测度及 MOPSO 占优所提方法的 SC 测度. “/”代表该处数据无实际意义 (后续表格同, 不再说明).

表 2　不同方法所选路径集包含的边数

程序	本文方法	所有可达边	MOPSO	贪心算法	整数规划	文献[35]方法
Triangle	18.0	18	18.0	18	18	18.0
Day	32.0	32	31.9	32	32	32.0
Numday	62.6	63	62.7	63	63	61.9
Totinfo	55.8	56	55.8	56	56	55.6
Replace	80.7	81	80.5	81	81	79.8
Space	1 192.4	1193	1 185.6	1193	1193	1 127.4
Flex	2 147.1	2148	2 141.5	2148	2148	1 952.5

表 3　不同方法所选路径集的包含率 (%)

程序	本文方法	MOPSO	贪心算法	整数规划	文献[35]方法
Triangle	100	100	100	100	100
Day	100	99.7	100	100	100
Numday	99.4	99.5	100	100	98.3
Totinfo	99.6	99.6	100	100	99.2
Replace	99.6	99.4	100	100	98.5
Space	99.9	99.4	100	100	94.5
Flex	100	99.7	100	100	90.9
平均值	99.8	99.6	100	100	97.3

表 4　不同方法所选路径集中路径条数

程序	本文方法	文献[21]方法	MOPSO	贪心算法	整数规划方法	文献[35]方法
Triangle	3.0	5.7	3.0	4.0	3.0	3.0
Day	12.3	13.9	12.5	13.9	12.0	12.0
Numday	16.5	21.5	16.9	19.2	16.0	16.6
Totinfo	19.4	20.8	19.8	20.8	19.0	19.1
Replace	24.7	32.3	25.9	28.8	24.0	22.6
Space	189.9	364.2	194.3	205.6	189.0	153.7
Flex	403.5	814.3	409.7	527.5	402	317.3

表 5　不同方法所选路径集的约减率 (%)

程序	本文方法	MOPSO	贪心算法	整数规划	文献[35]方法
Triangle	47.4	47.4	22.5	47.4	47.4
Day	11.5	10.1	11.5	13.7	13.7
Numday	23.3	21.4	14.1	23.6	22.8
Totinfo	6.7	4.8	6.7	8.7	8.1
Replace	23.5	19.8	14.4	25.7	30.0
Space	47.9	46.7	7.6	48.1	57.8
Flex	50.4	49.7	23.5	50.6	61.0
平均值	29.8	28.6	14.3	31.1	34.4

由图 4 和表 2–表 5: (1) 对于每一被测程序, 所提方法选择路径集的包含率均高达 99% 以上, 其中, 包含率最大的是 Triangle、Day 和 Flex, 均为 100%; 最小的是 Numday, 也达到 99.4%. 这 7 个被测程序的平均包含率为 99.8%, 这说明, 所提方法选择的路径集几乎能够包含全部可达边; (2) 相比于文献 [21] 生成的初始路径集, 对于每一被测程序, 所提方法选择的路径集均能大幅度减少路径数量, 其中, 约减率最大的是 Flex, 高达 50.4%; 最小的是 Totinfo, 达到 6.7%. 这 7 个被测程序的平均路径约减率为 29.8%; (3) 相比于同样是多目标算法的基于 MOPSO 选择的路径集, 对于每一被测程序, 所提方法选择路径集的包含率更高. 这说明, 所提方法能够比基于 MOPSO 的方法包含更多的边; 同时, 所提方法选择的路径集中路径数量更少, 约减率更高. 这说明, 所提方法得到的路径集更小; (4) 相比于贪心算法, 对于每一被测程序, 所提方法选择的路径集包含的边和包含率都略少于贪心算法, 这是因为, NSGA-II 具有随机性, 不能保证每次都能达到 100% 的边包含率. 所提方法选择的路径集的路径条数均小于贪心算法. 这表明, 所提方法选择的路径集, 能够大幅度减少冗余路径数量; (5) 相比于整数规划, 对于每一被测程序, 所提方法选择的路径集包含的边和包含率都略少于贪心算法, 这是因为, NSGA-II 具有随机性, 不能保证每次都能达到 100% 的边包含率. 所提方法选择的路径集的路径条数稍多于整数规划算法; (6) 相比于文献 [35] 选择的路径集, 对于每一被测程序, 所提方法选择路径集的包含率更高, 尤其对于较大的程序效果更明显; 所提方法选择

的路径集中路径条数虽然多于文献 [35] 方法, 但是文献 [35] 方法不能包含所有可达边, 也就不满足测试充分性准则.

由表 6: 一方面, 关于 HV 测度, 对于每一被测程序, 所提方法的平均 HV 值均大于基于 MOPSO 方法的平均 HV 值. 这说明, 在相同的迭代次数下, 所提方法得到 Pareto 前沿的分布性和收敛性更好. 另一方面, 对比它们之间的 SC 测度, 除 Triangle 外的其他程序中, 所提方法占优 MOPSO 所得最优解的比例均大于 MOPSO 占优所提方法所得最优解的比例. 这说明, 所提方法得到 Pareto 前沿的收敛性更好. 综合以上两方面, 所提方法的整体性能更好.

由表 7: 相比于贪心算法, 对于 Triangle、Day、Numday、Totinfo 和 Replace 这样的中小型程序, 所提方法的运行时间稍大于贪心算法, 但对于 Space 和 Flex 这样的大型程序, 所提方法的运行时间远小于贪心算法; 相比于 MOPSO, 可见两种方法的运行时间差距并不大, 但由以上几个表可知所提方法的整体性能更好; 相比于整数规划, 对于 Triangle 和 Day 这样的中小型程序, 所提方法的运行时间稍大于整数规划算法, 但对于 Space 和 Flex 这样的大型程序, 所提方法的运行时间远小于整数规划方法, 这也说明了整数规划方法虽然能够得到最优解, 但是计算时间会随着待求解规模的增大而成指数级增长. 因此, 在路径选择问题中, 对于中小型程序, 可以采用整数规划方法, 而对于工业大型程序, 采用所提方法是能够在可接受的时间内解决问题的.

表 6 不同多目标方法的 HV 值和 SC 值

程序	HV		SC	
	本文方法	MOPSO	本文方法 > MOPSO	MOPSO > 本文方法
Triangle	6	6	0	0
Day	167.9	163.7	0.075	0.013
Numday	394	374.2	0.30	0.14
Totinfo	436.2	425.4	0.38	0.17
Replace	775.5	786.1	0.50	0.24
Space	111 320.5	110 888.5	0.54	0.18
Flex	428 739.9	428 014.7	0.59	0.11

表 7 不同方法运行时间消耗 (s)

程序	本文方法	贪心算法	MOPSO	整数规划方法
Triangle	0.143	0.008	0.166	0.061
Day	1.174	0.04	1.137	0.622
Numday	1.177	0.113	1.206	1.357
Totinfo	1.238	0.111	1.255	1.343
Replace	1.365	0.418	1.401	2.538
Space	257.113	632.716	258.834	804.008
Flex	283.473	5 813.28	283.765	11 715.353

以上几点表明, 所提方法选择路径集能够包含所有可达边并且能大幅减少路径数量.

4.4.2 关于问题 3

为了回答问题 3, 首先, 由关于问题 1 和问题 2 的分析可以得出, MOPSO 得到的路径集没有所提方法得到的路径集好, 整数规划方法在大型程序的运行时间又过于长, 而文献 [35] 所提方法得到的路径集无法覆盖所有可达边, 所以在生成测试数据时没有将上述方法作为对比方法. 在实验时, 首先分别使用所提多目标方法、文献 [21] 所提方法和贪心算法得到目标路径集; 然后, 生成覆盖这些目标路径集的测试数据; 最后, 计算这 3 种方法生成测试数据的时间消耗. 此外, 针对记录的测试数据生成时间消耗, 基于 Mann-Whitney U 检验方法, 判断所提方法与其他方法的差异是否显著, 显著性水平取 0.05. 小于 0.05 代表两种方法的指标差异明显. 大于 0.05 代表两种方法的指标没有显著差异. 实验结果如表 8 和表 9 所列表中, 第 2–4 列为不同方法选择路径集用来生成测试数据的时间消耗和约减率, 约减率越大, 说明生成测试据的时间越短. 第 5 列为非参数假设检验结果.

由表 8: (1) 对于每一被测程序, 将所提方法选择的路径集作为待覆盖路径集, 生成测试数据的时间消耗均小于将文献 [21] 生成的基路径集作为待覆盖路径生成测试数据的时间消耗, 其中, 约减率最小的是 Totinfo, 为 13.5%; 约减率最大的是 Space, 为 66.8%, 平均约减率为 42.9%. (2) 对于每一被测程序, 所提方法与文献 [21] 生成的基路径集用来生成测试数据的时间消耗均有显著差异. 这些说明, 使用所提方法能缩短测试数据生成所需的时间, 从而提高测试数据生成效率.

由表 9: (1) 对于每一被测程序, 将所提方法选择的路径集作为待覆盖路径集, 生成测试数据的时间消耗均小于将基路径集作为待覆盖路径生成测试数据的时间消耗, 其中, 约减率最小的是 Space, 为 8.9%; 约减率最大的是 Flex, 为 30.3%, 平均约减率为 19.4%. (2) 对于每一被测程序, 所提方法与贪心算法选择的路径集用来生成测试数据的时间消耗均有显著差异. 说明使用所提方法能缩短测试数据生成所需的时间, 从而提高测试数据生成效率.

表 8 本文方法与文献 [21] 的结果比较

程序	时间消耗 (s)		约减率 (%)	非参数假设检验结果
	本文	文献[21]		
Triangle	0.0069	0.0143	51.7	<0.05
Day	0.1724	0.2085	17.3	<0.05
Numday	0.4632	1.0594	59.6	<0.05
Totinfo	43.0882	49.8391	13.5	<0.05
Replace	223.6504	329.4659	32.1	<0.05
Space	13498.1419	40655.7683	66.8	<0.05
Flex	41788.4268	102470.7840	59.2	<0.05
平均值	—	—	42.9	—

表 9 本文方法与贪心算法的结果比较

程序	时间消耗 (s)		约减率 (%)	非参数假设检验结果
	本文	贪心算法		
Triangle	0.0069	0.0084	17.9	<0.05
Day	0.1724	0.2468	30.1	<0.05
Numday	0.4632	0.5654	18.1	<0.05
Totinfo	43.0882	48.6365	11.4	<0.05
Replace	223.6504	277.5742	19.4	<0.05
Space	13498.1419	14815.1413	8.9	<0.05
Flex	41788.4268	59937.5797	30.3	<0.05
平均值	—	—	19.4	—

此外, 为了验证所提方法中优化求解的代价能够少于原来 (未约简时) 多出的测试代价, 我们采用了文献 [43] 中提出的遗传算法来为程序 Flex 中约简的路径生成测试数据. 关于遗传算法的参数设置, 请参考第 4.3 节. 当进行该任务时, 根据对应于个体的测试数据执行程序的时间消耗为 0.012 s. 然后, 一个规模为 100 的种群执行程序要消耗 1.2 s. 我们假设种群进化了 100 代, 生成了期望的测试数据 (这是一个高效率的情况), 总时间消耗为 120 s. 请注意, 上述时间消耗是针对仅覆盖一条路径而实现的. 对于包含大小为 411 的路径集的程序, 生成所有测试数据需要 49 320 s. 对于更复杂的程序, 测试成本会更大, 这表明需要适当的方法来大大降低测试成本. 相比之下, 本文提出的方法虽然在求解多目标模型过程中需要花费一些时间, 但只需 83.473 s, 远小于对冗余路径生成测试数据的时间. 这表明本文对测试路径集进行约简是有必要的, 能够提高测试效率.

综合以上 3 个问题的实验结果与分析可知: (1) 应用本文建立的模型能够在保证测试覆盖率的情况下, 有效减少路径集包含的路径条数; (2) 针对路径选择优化模型设计的多目标方法, 能够有效降低测试数据生成的时间消耗, 提高测试效率.

5 对有效性的威胁

(1) 内部威胁: 个体群优化的参数设置会影响实验结果. 参数不同会影响实验的运行时间及结果. 为了减少这类威胁, 个体群优化的参数参照文献 [42]. 从实验结果可以看出, 上述参数设值是合理的.

(2) 外部威胁: 所选的被测程序影响实验结论. 如果所选程序不具有代表性或者不足, 那么, 运行时间及实验结果就很难直接推广到其他程序中. 为了尽可能减轻这类威胁, 我们在实验中选择 7 个具有不同代码行数的程序, 其中行数最多的一个程序为 10 459 行.

此外, 时间消耗是反映所提方法效率的重要指标. 实验中, 以秒的形式描述时间消耗, 该值与实现环境的配置, 如处理器数量与类型等密切相关. 显然, 在完成相同任务时, 具有不同配置的环境将具有不同的时间消耗. 为了缓解该威胁, 在相同环境中, 运行 10 次相同的任务, 并记录时间消耗, 基于此计算平均时间消耗.

6 总 结

本文研究路径覆盖测试中目标路径的选择问题, 将路径选择问题建模为一个多目标优化问题, 并采用多目标进化算法求解, 从路径选择的角度降低路径测试的时间消耗, 提高路径测试效率. 所提方法首先以多条路径作为决策变量, 以决策变量包含的边数和路径数为两个目标, 建立多目标优化模型; 然后, 采用多目标进化算法求解该模型, 得到 Pareto 最优解集, 基于此, 选择目标路径集.

为了评估本文方法的性能, 将提出的方法应用于 7 个基准和工业程序测试, 并与文献 [21] 所提方法、MOPSO、贪心算法、整数规划及文献 [35] 所提方法比较. 实验结果表明, 所提方法能够在保证测试覆盖率的前提下, 有效减少路径集包含的路径条数, 降低测试数据生成的时间消耗, 从而提高测试数据生成效率.

需要说明的是, 本文在选择目标路径集时, 仅考虑路径集包含的边数和路径条数, 没有考虑这些可达路径的覆盖难度. 一种可能的情况是, 虽然路径条数少, 但如果这些路径难以覆盖, 那么, 生成覆盖这些路径的测试数据也比

较困难, 从而增加了路径覆盖测试数据生成的时间消耗. 鉴于此, 在后续的研究中, 有必要进一步结合路径的覆盖难度, 生成条数少且容易覆盖的目标路径集, 以提高路径测试数据生成效率. 同时, 关于路径测试中揭错率和缺陷预测问题, 都是需要进一步考虑的问题. 此外, 对于并行程序的路径测试, 如何选择目标路径集, 也是需要进一步研究的问题.

References:

[1] Gonçalves WF, de Almeida CB, de Araújo LL, Ferraz MS, Xandú RB, de Farias I. The influence of human factors on the software testing process: The impact of these factors on the software testing process. In: Proc. of the 12th Iberian Conf. on Information Systems and Technologies. Lisbon: IEEE, 2017. 1–6. [doi: 10.23919/CISTI.2017.7975873]

[2] Myers GJ, Badgett T, Thomas TM, Sandler C. The Art of Software Testing. 2nd ed., Hoboken: John Wiley & Sons, 2004.

[3] Wang Y. Research on inner-function path generation and infeasible path detection [MS. Thesis]. Beijing: Beijing University of Posts and Telecommunications, 2016 (in Chinese with English abstract).

[4] Aleti A, Grunske L. Test data generation with a Kalman filter-based adaptive genetic algorithm. Journal of Systems and Software, 2015, 103: 343–352. [doi: 10.1016/j.jss.2014.11.035]

[5] Jiang SJ, Shi JJ, Zhang YM, Han H. Automatic test data generation based on reduced adaptive particle swarm optimization algorithm. Neurocomputing, 2015, 158: 109–116. [doi: 10.1016/j.neucom.2015.01.062]

[6] Gong DW, Zhang Y. Novel evolutionary generation approach to test data for multiple paths coverage. Acta Electronica Sinica, 2010, 38(6): 1299–1304 (in Chinese with English abstract).

[7] Wei QJ, Li YJ, Zhang YH. A new method of evolutionary testing for path coverage. In: Proc. of the 2018 IEEE Int'l Conf. on Software Quality, Reliability and Security Companion. Lisbon: IEEE, 2018. 79–86. [doi: 10.1109/QRS-C.2018.00028]

[8] Ding R, Dong HB, Zhang Y, Feng XB. Fast automatic generation method for software testing data based on key-point path. Ruan Jian Xue Bao/Journal of Software, 2016, 27(4): 814–827 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4971.htm [doi: 10.13328/j.cnki.jos.004971]

[9] Sun BC, Wang JX, Gong DW, Tian T. Scheduling sequence selection for generating test data to cover paths of MPI programs. Information and Software Technology, 2019, 114: 190–203. [doi: 10.1016/j.infsof.2019.07.002]

[10] Liao WZ. Test data generation based on automatic division of path. Act Electron Sinica, 2016, 44(9): 2254−2261 (in Chinese with English abstract). [doi: 10.3969/j.issn.0372-2112.2016.09.034]

[11] Liu FQ, Huang H, Hao ZF. Evolutionary algorithm with convergence speed controller for automated software test data generation problem. In: Proc. of the 2017 IEEE Congress on Evolutionary Computation. San Sebastian: IEEE, 2017. 869–875. [doi: 10.1109/CEC.2017.7969400]

[12] Huang H, Liu FQ, Zhuo XY, Hao ZF. Differential evolution based on self-adaptive fitness function for automated test case generation. IEEE Computational Intelligence Magazine, 2017, 12(2): 46–55. [doi: 10.1109/MCI.2017.2670462]

[13] Panichella A, Kifetew FM, Tonella P. Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets. IEEE Trans. on Software Engineering, 2018, 44(2): 122–158. [doi: 10.1109/TSE.2017.2663435]

[14] Arrieta A, Wang S, Markiegi U, Sagardui G, Etxeberria L. Employing multi-objective search to enhance reactive test case generation and prioritization for testing industrial cyber-physical systems. IEEE Trans. on Industrial Informatics, 2018, 14(3): 1055–1066. [doi: 10.1109/TII.2017.2788019]

[15] Han H. Research on detection of infeasible basis paths based on association analysis [MS. Thesis]. Xuzhou: China University of Mining and Technology, 2014 (in Chinese with English abstract).

[16] McCabe TJ. Structured testing: A software testing methodology using the cyclomatic complexity metric. Washington: U.S. Dept. of Commerce, National Bureau of Standards, 1982.

[17] Qian L. Methodology on qualitative simulation modeling of software reliability based on chaos theory. In: Proc. of the 5th IEEE Int'l Conf. on Software Engineering and Service Science. Beijing: IEEE, 2014. 99–104. [doi: 10.1109/ICSESS.2014.6933522]

[18] Zolotov V, Xiong JJ, Fatemi H, Visweswariah C. Statistical path selection for at-speed test. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2010, 29(5): 749–759. [doi: 10.1109/TCAD.2010.2043570]

[19] Wijayasiriwardhane TK, Wijayarathna PG, Karunarathna DD. An automated tool to generate test cases for performing basis path testing. In: Proc. of the 2011 Int'l Conf. on Advances in ICT for Emerging Regions. Colombo: IEEE, 2011. 95–101. [doi: 10.1109/ICTer.2011.6075032]

[20] Wang G, Jing XN, Wang YJ. The application of improved McCabe method in basis path test. Journal of Harbin University of Science and

Technology, 2010, 15(1): 48–51 (in Chinese with English abstract). [doi: 10.15938/j.jhust.2010.01.021]

[21] Ghiduk AS. Automatic generation of basis test paths using variable length genetic algorithm. Information Processing Letters, 2014, 114(6): 304–316. [doi: 10.1016/j.ipl.2014.01.009]

[22] Xia L, Shang W, Wang ZX. The research of path testing in doftware business process based on genetic algorithm. Computer Engineering & Software, 2019, 40(4): 133–139 (in Chinese with English abstract). [doi: 10.3969/j.issn.1003-6970.2019.04.029]

[23] Srivastava PR, Baby K, Raghurama G. An approach of optimal path generation using ant colony optimization. In: Proc. of 2009 TENCON IEEE Region 10 Conf. Singapore: IEEE, 2010. 1–6. [doi: 10.1109/TENCON.2009.5396088]

[24] Bertolino A, Marré M. Automatic generation of path covers based on the control flow analysis of computer programs. IEEE Trans. on Software Engineering, 1994, 20(12): 885–899. [doi: 10.1109/32.368137]

[25] Du QF, Dong X. An improved algorithm for basis path testing. In: Proc. of the 2011 Int'l Conf. on Business Management and Electronic Information. Guangzhou: IEEE, 2011. 175–178. [doi: 10.1109/ICBMEI.2011.5920422]

[26] Liu Y, Zeng M, Zhu L, Chen JF, Yan JW. Automatic test sequences generation technology based on data flow rules. Microelectronics & Computer, 2005, 22(5): 131–135 (in Chinese with English abstract). [doi: 10.19304/j.cnki.issn1000-7180.2005.05.033]

[27] Zhang GM, Chen R, Li XW, Han CY. The automatic generation of basis set of path for path testing. In: Proc. of the 14th Asian Test Symp. Calcutta: IEEE, 2005. 46–51. [doi: 10.1109/ATS.2005.106]

[28] Chen Y, Wang AB, Wang JJ, Liu L, Song YZ, Ha QH. Automatic test transition paths generation approach from EFSM using state tree. In: Proc. of the 2018 IEEE Int'l Conf. on Software Quality, Reliability and Security Companion. Lisbon: IEEE, 2018. 87–93. [doi: 10.1109/QRS-C.2018.00029]

[29] Feng LH. Based on set evolution algorithm of EFSM model feasible test sequence suite generation [MS. Thesis]. Beijing: Beijing University of Chemical Technology, 2017 (in Chinese with English abstract).

[30] Wang SL, Wang YW, Gong YZ. Path choses based on interval arithmetic for unit coverage testing. Journal of Tsinghua University (Science and Technology), 2011, 51(S1): 1402–1406, 1413 (in Chinese with English abstract). [doi: 10.16511/j.cnki.qhdxxb.2011.s1.006]

[31] Li QC, Wang YW, Gao WL. A method of path choosing for modified condition/decision coverage testing. Sciencepaper Online. 2010. https://www.edu.cn/ke_yan_yu_fa_zhan/gao_xiao_cheng_guo/lun_wen_zhai_bao/201011/t20101112_538329.shtml (in Chinese).

[32] Gao JX, Lumetta SS. Loop path reduction by state pruning. In: Proc. of the 33rd IEEE/ACM Int'l Conf. on Automated Software Engineering. Montpellier: IEEE, 2018. 838–843. [doi: 10.1145/3238147.3240731]

[33] Wu C, Gong DW, Yao XJ. Selection of paths for regression testing based on branch coverage. Ruan Jian Xue Bao/Journal of Software, 2016, 27(4): 839–854 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4975.htm [doi: 10.13328/j.cnki.jos.004975]

[34] Feng JJ. Research on complex networks based the least testing paths generation [MS. Thesis]. Dalian: Dalian University of Technology, 2010 (in Chinese with English abstract).

[35] Prema P, Ramadoss B, Balasundaram SR. An approach for reduced test path set selection. In: Proc. of the 2013 Int'l Conf. on Advances in Computing, Communications and Informatics. Mysore: IEEE, 2013. 191–196. [doi: 10.1109/ICACCI.2013.6637169]

[36] Hutchins M, Foster H, Goradia T, Ostrand T. Experiments on the effectiveness of dataflow-and control-flow-based test adequacy criteria. In: Proc. of the 16th Int'l Conf. on Software Engineering. Sorrento: IEEE, 1994. 191–200. [doi: 10.1109/ICSE.1994.296778]

[37] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation, 2002, 6(2): 182–197. [doi: 10.1109/4235.996017]

[38] Zhong H, Zhang L, Mei H. An experimental study of four typical test suite reduction techniques. Information and Software Technology, 2008, 50(6): 534–546. [doi: 10.1016/j.infsof.2007.06.003]

[39] Deng Q, Gao JJ, Ge DD, He SM, Jiang B, Li XC, Wang ZZ, Yang CL, Ye YY. Modern optimization theory and applications. Scientia Sinica Mathematica, 2020, 50(7): 899–968 (in Chinese with English abstract). [doi: 10.1360/SSM-2020-0035]

[40] Offutt AJ, Pan J. Automatically detecting equivalent mutants and infeasible paths. Software Testing, Verification and Reliability, 1997, 7(3): 165–192. [doi: 10.1002/(SICI)1099-1689(199709)7:3<165::AID-STVR143>3.0.CO;2-U]

[41] Mouchawrab S, Briand LC, Labiche Y, Di Penta M. Assessing, comparing, and combining state machine-based testing and structural testing: A series of experiments. IEEE Trans. on Software Engineering, 2011, 37(2): 161–187. [doi: 10.1109/TSE.2010.32]

[42] Song JX. Research on test case generation method for mobile testing [MS. Thesis]. Changchun: Jilin University, 2019 (in Chinese with English abstract).

[43] McMinn P. Evolutionary search for test data in the presence of state behavior [Ph.D. Thesis]. Sheffield: University of Sheffield, 2005.

[44] Clerc M. Discrete particle swarm optimization, illustrated by the traveling salesman problem. In: Onwubolu GC, Babu BV. New Optimization Techniques in Engineering. Berlin Heidelberg: Springer, 2004. 219–239. [doi: 10.1007/978-3-540-39930-8_8]

[45] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Trans. on Evolutionary Computation, 1999, 3(4): 257–271. [doi: 10.1109/4235.797969]

附中文参考文献:

[3] 王毅. 函数间路径生成与不可达判定技术研究[硕士学位论文]. 北京: 北京邮电大学, 2016.
[6] 巩敦卫, 张岩. 一种新的多路径覆盖测试数据进化生成方法. 电子学报, 2010, 38(6): 1299–1304.
[8] 丁蕊, 董红斌, 张岩, 冯宪彬. 基于关键点路径的快速测试用例自动生成方法. 软件学报, 2016, 27(4): 814–827. http://www.jos.org.cn/1000-9825/4971.htm [doi: 10.13328/j.cnki.jos.004971]
[10] 廖伟志. 基于路径自动分割的测试数据生成方法. 电子学报, 2016, 44(9): 2254–2261. [doi: 10.3969/j.issn.0372-2112.2016.09.034]
[15] 韩寒. 基于关联分析的不可达基路径检测技术研究[硕士学位论文]. 徐州: 中国矿业大学, 2014.
[20] 王冠, 景小宁, 王彦军. 基本路径测试中的McCabe算法改进与应用. 哈尔滨理工大学学报, 2010, 15(1): 48–51. [doi: 10.15938/j.jhust.2010.01.021]
[22] 夏良, 商伟, 王兆星. 基于遗传算法的软件业务流程测试路径的研究. 软件, 2019, 40(4): 133–139. [doi: 10.3969/j.issn.1003-6970.2019.04.029]
[26] 刘勇, 曾明, 朱利, 陈继峰, 严建伟. 基于数据流的软件测试序列自动生成技术研究. 微电子学与计算机, 2005, 22(5): 131–135. [doi: 10.19304/j.cnki.issn1000-7180.2005.05.033]
[29] 冯利航. 基于集合进化算法的EFSM模型可行测试序列集生成[硕士学位论文]. 北京: 北京化工大学, 2017.
[30] 王思岚, 王雅文, 宫云战. 单元覆盖测试中基于区间运算的路径选择. 清华大学学报(自然科学版), 2011, 51(S1): 1402–1406, 1413. [doi: 10.16511/j.cnki.qhdxxb.2011.s1.006]
[31] 李青翠, 王雅文, 高文玲. 一种修正条件判定覆盖测试的路径选取方法. 中国科技论文在线. 2010. https://www.edu.cn/ke_yan_yu_fa_zhan/gao_xiao_cheng_guo/lun_wen_zhai_bao/201011/t20101112_538329.shtml
[33] 吴川, 巩敦卫, 姚香娟. 基于分支覆盖的回归测试路径选择. 软件学报, 2016, 27(4): 839–854. http://www.jos.org.cn/1000-9825/4975.htm [doi: 10.13328/j.cnki.jos.004975]
[34] 冯金金. 基于复杂网络的软件最小测试路径生成研究[硕士学位论文]. 大连: 大连理工大学, 2010.
[39] 邓琪, 高建军, 葛冬冬, 何斯迈, 江波, 李晓澄, 王子卓, 杨超林, 叶荫宇. 现代优化理论与应用. 中国科学: 数学, 2020, 50(7): 899–968. [doi: 10.1360/SSM-2020-0035]
[42] 宋佳垿. 面向移动端测试的测试用例生成方法研究[硕士学位论文]. 长春: 吉林大学, 2019.

杜莹(1996－), 女, 硕士, 主要研究领域为基于搜索的软件工程.

田甜(1987－), 女, 博士, 副教授, CCF 专业会员, 主要研究领域为程序分析与测试, 智能优化.

孙百才(1990－), 男, 博士, CCF 专业会员, 主要研究领域为基于搜索的软件工程, 数据驱动进化优化.

姚香娟(1975－), 女, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为进化测试.

巩敦卫(1970－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为基于搜索的软件工程.