一种基于优先级的迭代划分测试方法

章晓芳[1),2)] 章宗长[1)] 谢晓园[3)] 周谊成[1)]

[1)](苏州大学计算机科学与技术学院 江苏 苏州 215006)

[2)](南京大学计算机软件新技术国家重点实验室 南京 210023)

[3)](武汉大学软件工程国家重点实验室 武汉 430072)

摘 要 随机测试和划分测试是两种重要的测试方法,关于两者在失效检测能力和效率方面的比较一直是软件测试领域的研究热点之一.适应性随机测试是对随机测试的一种增强,通过实现测试用例在输入域上的均匀分布,提高了随机测试的失效检测能力.该文从划分测试出发,借鉴了均匀分布的思想,提出了一种基于优先级的迭代划分测试方法(Iterative Partition Testing based on Priority Sampling,IPT-PS).首先迭代划分输入域并选取划分后子域的中心点作为待执行的测试用例,随后采取优先级策略,将待执行的测试用例分为 3 种不同优先等级并依次执行.迭代划分和中心采样仅需要已知输入域的空间信息,优先级执行则考虑了测试用例的不同空间特性,上述 3 种操作均仅需要很少的时间开销并力求实现测试用例在输入域上的均匀分布,以提高失效检测能力.该文通过理论分析给出了 IPT-PS 检测出对应失效所需测试用例数量的上界,并通过一系列实验结果表明:IPT-PS 在仅使用接近随机测试时间开销的情况下,可以获得与适应性随机测试相近甚至更好的失效检测能力,是一种高效的测试方法.

关键词 软件测试;划分测试;随机测试;适应性随机测试;测试用例生成;失效率;*F*-度量

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2016.02307

An Approach of Iterative Partition Testing Based on Priority Sampling

ZHANG Xiao-Fang[1),2)] ZHANG Zong-Zhang[1)] XIE Xiao-Yuan[3)] ZHOU Yi-Cheng[1)]

[1)](*School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu* 215006)

[2)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023)

[3)](*State Key Laboratory of Software Engineering, Wuhan University, Wuhan* 430072)

Abstract Random testing and partition-based testing are two important test case generation methods. Comparisons on the efficiency and effectiveness between these two methods have been a popular research area. As an enhanced version of random testing, Adaptive Random Testing (ART) aims to evenly spread test cases within the input domain, in order to achieve better failure detection effectiveness. This paper follows the intuition of ART that high diversity is helpful in detecting failures, and proposes a novel algorithm, named Iterative Partition Testing based on Priority Sampling (IPT-PS). IPT-PS iteratively divides the input domain into grids, and selects the center point of each grid as test case. Priority-based execution strategy is then applied on newly generated test cases in each round of iteration. Iterative partition and fixed-center-point sampling only require information of the input domain, while priority-based execution considers different spatial characteristics of test cases. All these three steps need trivial time cost, and help to sample test cases much more even than traditional fixed-size partition testing, such that better

收稿日期:2015-09-10;在线出版日期:2016-03-26.本课题得到国家自然科学基金(61103045,61502329,61502323,61572375)、软件新技术与产业化协同创新中心部分资助.**章晓芳**,女,1980 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为软件分析与测试、错误定位、强化学习等.E-mail:xfzhang@suda.edu.cn.**章宗长**(通信作者),男,1985 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为部分感知的马尔科夫决策过程、强化学习、多 agent 系统等.E-mail:zzzhang@suda.edu.cn.**谢晓园**,女,1983 年生,博士,教授,中国计算机学会(CCF)会员,主要研究方向为软件分析与测试、错误定位、调试、基于搜索的软件工程等.**周谊成**,男,1990 年生,硕士研究生,主要研究方向为软件工程、强化学习等.

failure detection effectiveness can be achieved. We theoretically prove the upper bound of the effectiveness for our method, and conduct comprehensive empirical analysis. Our experimental results show that IPT-PS can achieve effectiveness as high as ART, but with time cost as low as random testing.

Keywords software testing; partition testing; random testing; adaptive random testing; test case generation; failure rate; *F*-measure

1 引 言

软件测试是软件质量保证中的一种重要方法，受到了学术界和工业界的广泛关注.在软件测试的过程中，一项重要的工作是在程序的输入域中选择测试用例，用于执行待测程序以发现待测程序中的错误.由于对软件整个输入域进行穷尽测试往往是不现实的，因此如何选择尽可能少的测试用例来尽快发现程序错误变得尤为重要.

在现有的众多测试方法中，随机测试（Random Testing，RT）和划分测试（Partition Testing，PT）是两种经典的方法.随机测试服从某种概率分布（通常是均匀分布），从整个输入域中逐个选择测试用例，输入域没有进行任何划分[1-2]；而划分测试则首先将输入域划分为一定数量的不相交的子域，再从每个子域中选取一个或多个的测试用例[3].

随机测试具有简单有效、易于理解和实现等优点，然而，由于随机测试过程中没有利用其他任何有用的信息，其失效检测能力往往不是很理想.Chen 等人[4]提出了适应性随机测试方法（Adaptive Random Testing，ART），是一种改进的随机测试方法，通过实现测试用例在输入域上的均匀分布，来提高随机测试的失效检测能力，但该方法也相应地带来大量额外的计算开销.

划分测试通常基于相同的功能或属性进行子域的划分，希望能获得理想的划分方式使得每个子域是同质的，即同一子域中的所有输入均为正确的输入或导致程序失效的输入.在获得有效划分的基础上，划分测试用于选择测试用例的开销并不大，然而，前期的子域划分却需要耗费大量额外的时间开销.因此，关于随机测试和划分测试在失效检测能力和效率方面的比较一直是软件测试领域的研究热点之一[3,5-6].

本文从划分测试出发，借鉴了 ART 均匀分布的思想，给出了迭代划分测试方法（Iterative Partition Testing，IPT）的实现框架，并进一步提出了一种基于优先级的迭代划分测试方法（Iterative Partition Testing based on Priority Sampling，IPT-PS）.该方法的基本思想是：通过不断地迭代划分输入域并选取划分后子域的中心点作为待执行的测试用例，随后采取优先级策略，将待执行的测试用例分为 3 种不同优先等级并依次执行.中心采样和迭代划分仅需要知道输入域的空间信息.优先级执行则考虑了待执行测试用例的不同空间特性，通过约束测试用例执行的先后次序使得测试用例在两轮迭代间的分布更加均匀.

IPT-PS 的中心采样、迭代划分和优先级执行这 3 种操作仅需知道输入域的空间信息，且仅耗费很少的时间开销，弥补了传统划分测试通常需要知道额外信息或往往耗费大量开销用于划分子域的不足.本文的方法力求快速实现测试用例在输入域上的均匀分布，以提高失效检测能力.本文通过理论分析指出：IPT-PS 可以检测出对应失效所需测试用例数量的上界.实验结果表明，IPT-PS 方法在仅使用接近 RT 时间开销的情况下，可以获得与 ART 相近甚至更好的失效检测能力.因此，我们的方法将在失效检测能力和时间开销上获得良好的平衡，是一种高效的测试方法.

本文第 2 节将介绍随机测试以及划分测试的相关概念和研究现状；第 3 节提出 IPT-PS 的框架及算法描述，并给出对应的理论分析结果；第 4 节通过一系列实验验证理论结果，并对比多种划分测试以及随机测试方法的失效检测能力和时间开销，探讨 IPT-PS 的适用性；最后是总结和未来工作.

2 相关工作

为便于后续讨论，首先给出一些相关术语的描述和解释.

定义 1. 失效率（failure rate）[4].在测试过程中，假定待测程序的输入域大小为 D，能够引发失效

的输入域大小为 F，则失效率 θ 为失效输入域占整个输入域的比例，即 $\theta=F/D$.

定义 2.　失效模式(failure pattern)[7]. 指失效输入域的几何特性及其在输入域上的分布情况. 具体而言，包括失效输入域的形状、大小、方位、位置等信息.

Chan 等人[7]总结了 3 种失效模式，分别为块状模式、条状模式和点状模式. 显然，给定待测程序，其失效率和失效模式是确定的，但在测试完成前通常是未知的.

定义 3.　F-度量(F-measure)[4]. 指检测到第 1 个失效时所使用的测试用例数量的期望值. F-measure 可用于表征测试用例集的失效检测能力，即测试用例集的有效性.

基于上述定义，理论上当采用放回策略选择测试用例时，随机测试的 F-measure 将等于 $1/\theta$.

2.1　随机测试

随机测试是一种重要的测试方法，它通常等概率地从输入域中逐个选择测试用例. 随机测试不仅可以独立使用，还可以是其他众多测试方法中的重要组成部分. 随机测试已在工业界得到广泛应用，特别是在规约说明和源代码不可获得或不完整的场景下，随机测试仍然可行. 此外，随机测试具有重要的统计意义，可以从理论上分析其失效检测能力，受到了学术界的广泛关注.

为了提高随机测试的失效检测能力，Chen 等人[4]提出了 ART 方法. 该方法的提出基于以下观察：已有众多实验研究表明引发失效的输入往往聚集在某一连续的区域内，于是，不会引发失效的输入也同样形成了一片连续的区域[8]. 因此，如果已执行的测试用例未能引发程序失效，那么接下来新生成的测试用例就应该与所有已执行且未能引发程序失效的测试用例的距离尽可能的远，从而使生成的测试用例尽可能均匀地分布在整个输入域，以便快速检测到失效区域.

固定规模的候选测试用例集的 ART 算法(Fixed Sized Candidate Set ART，FSCS-ART)是基于距离计算的一个经典算法[4]. 在选择下一个测试用例时，首先随机生成固定数量的候选测试用例，对于每个候选测试用例 c_j，找出 c_j 与已执行测试用例集中最靠近的测试用例，获得这两个测试用例之间的距离 d_j. 拥有最大距离 d_j 的候选测试用例将作为下一个待执行的测试用例. 已有实验结果表明：相比 RT，FSCS-ART 在失效检测能力方面有显著的提高. 然而，由于 FSCS-ART 在产生测试用例时需要逐一计算候选集中的测试用例与已执行的测试用例之间的距离，所以算法的计算开销较大.

在此基础上，众多研究学者提出了一系列算法来实现测试用例在输入域上的均匀分布[9-21]，同时，为降低 ART 算法生成测试用例的计算开销，提出了一些通用的技术，例如划分(partitioning)技术[15-17]、过滤(filtering)技术[18]、镜像(mirroring)技术[19]、聚类(clustering)技术[20]、遗忘(forgetting)技术[21]等. 特别地，研究学者引入划分的思想，提出了一些基于划分的 ART 算法，例如基于随机划分的[15]、基于二分划分的[15]、基于迭代划分的[16]、基于格划分[17]的 ART 算法.

众多的 ART 实现算法具有不同的失效检测能力、不同的计算开销和各自的适用场景[22-24]. Mayer 等人[24]通过一系列实验比较了现有的 13 种 ART 实现算法，并指出：基于距离计算的 ART 实现算法，如 FSCS-ART 的失效检测能力较强，基于迭代划分的 ART 算法(ART through Iterative Partitioning，IP-ART)的计算开销较小.

IP-ART 算法的基本思想是：以二维输入域为例，首先将输入域划分为 $p\times p$ 个格子，每次从可选的格子集合中随机选定一个格子，并在该格子中随机生成测试用例. 可选的格子集合不包含已执行的测试用例所在的格子以及与该格子相邻的所有格子. 当可选的格子集合为空时，将输入域进一步划分为 $(p+1)\times(p+1)$ 个格子[16]. IP-ART 通过划分来选定下一个测试用例的可选区域，避免了大量的距离计算开销，并保持了较高的失效检测能力.

2.2　划分测试

划分测试首先将输入域分割成多个不相交的子域，随后在每个子域中选择测试用例. 在理想的划分情况下，每个子域中的所有元素或者都能检测出失效，或者都无法检测出失效，此时对于每个子域仅需要选择其中任意一个元素进行测试即可. 然而，目前没有一种方法能够确保获得这样理想的划分，除非每个子域中仅包含一个元素，然而这样的划分测试将退化为穷尽测试[25]；此外，在划分测试中，对于输入域的划分往往是需要额外时间开销的，因此，需要衡量划分测试的划分代价是否可接受.

众多研究学者将划分测试和随机测试相比较. Weyuker 等人[26]提出了一种同等大小-同等抽样数的划分测试策略(Equal-Size-Equal-Sampling Strategy，ESESS)，该策略首先将输入域划分为相等大小的子

域集合,随后有放回地从每个子域中随机选取相同数量的测试用例.当每个输入具有相同的失效概率,ESESS 和 RT 选取相同数量的测试用例时,ESESS 检测出至少一个失效的概率(即 P-measure)将等于或高于 RT 的 P-measure.在此基础上,Chen 等人[27]提出了更为通用的比例抽样策略(Proportional Sampling Strategy,PSS),要求从每个子域中有放回地随机选取的测试用例数应与该子域的大小成固定比例,并证明了 PSS 是获得不低于 RT 的 P-measure的充分必要条件.当仅已知输入域的空间信息,难以进行其他有效划分时,PSS 是一种可行的划分测试方法.

Cai 等人将划分测试与随机测试相结合,提出了随机划分策略(Random Partition Testing,RPT)[28].该策略首先服从某一概率分布地随机选择一个划分后的子域,随后从选定的子域中服从等概率分布地随机选择一个测试用例.为改进 RPT 的失效检测能力,Cai 等人[29]提出了适应性测试(Adaptive Testing,AT)策略,将待测对象建模为一个受控的马尔科夫链,通过在线收集测试历史信息来提高失效检测能力.然而,AT 的反馈机制需要预先收集测试历史信息且需要耗费大量的计算开销用于测试用例的选择.近期,Cai 等人[30]将 AT 与 RPT 相结合,力求实现失效检测能力和时间开销方面的平衡.

尽管现有众多测试方法,包括 PT、ART 等都试图改进 RT 的失效检测能力,Chen 等人在文献[31]中给出了一个理论证明:除非已知失效输入域的位置,否则即便已知失效输入域的形状、大小、方位等信息(事实上我们往往无法事先获知这些信息),没有一种测试策略可以保证将其 F-measure 降低至 RT F-measure 的 50%.因此,50%的 RT F-measure 将是最优的测试用例选择策略的失效检测能力的上界.已有实验研究结果表明:ART 的失效检测能力已经非常接近这个最优的上界[25].

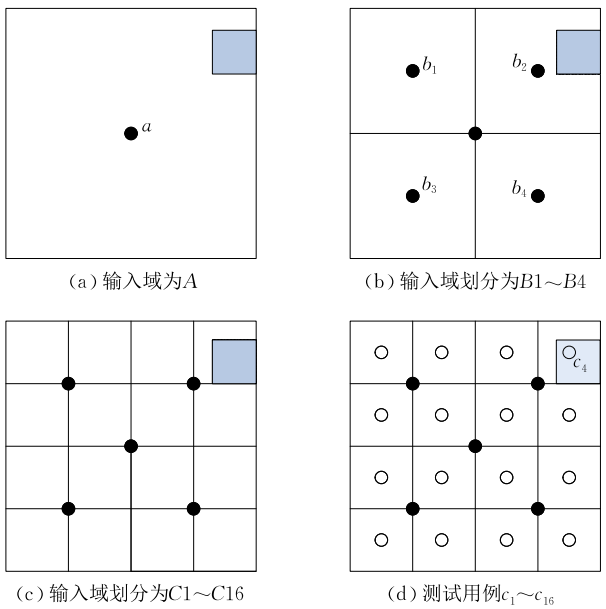
基于已有相关工作和分析,本文从划分测试出发,借鉴了 ART 均匀分布的思想,给出了一种迭代划分测试方法,该方法是一种基于输入域空间信息的迭代测试划分方法,较之其他基于功能、路径或风险等信息的划分测试方法,该方法所需的输入域空间信息容易获得且不需要做额外的预处理,极大地减小了划分测试在预处理方面的开销.同时,本文的方法借鉴了 ART 均匀分布的思想,希望能在仅使用接近 RT 时间开销的基础上,获得与 ART 相近甚至更好的失效检测能力.

3 迭代划分测试方法

本节首先通过一个简单的例子来展示迭代划分测试方法 IPT 的过程,随后给出 IPT 的实现框架,在此基础上提出了 IPT-PS 方法,最后从理论上分析了 IPT 和 IPT-PS 的失效检测能力.

3.1 一个例子

为便于理解,假定待测程序的输入域为二维矩形区域,失效域为块状模式.如图 1(a)~(d)所示,IPT 首先选择输入域 A 的中心点 a 作为测试用例执行,若 a 未能命中失效区域,则依据 a 将输入域划分为 4 个子区域 $B1$~$B4$,分别选取 $B1$~$B4$ 的中心点 b_1~b_4 作为测试用例执行;若 b_1~b_4 均未能命中失效区域,则依据 b_1~b_4 将输入域划分为 16 个子区域 $C1$~$C16$,分别选取 $C1$~$C16$ 的中心点 c_1~c_{16} 作为测试用例执行;此时,右上方子区域 $C4$ 的中心点 c_4 将命中失效区域,即检测出失效.反之,若 c_1~c_{16} 均未能命中失效区域,则将迭代进行下一轮划分,产生 64 个子区域.



(a)输入域为A (b)输入域划分为$B1$~$B4$

(c)输入域划分为$C1$~$C16$ (d)测试用例c_1~c_{16}

图 1 2 次划分产生 16 个子域的示意图

3.2 迭代划分测试方法(IPT)

由 3.1 节的示例可知:IPT 将包含迭代划分、中心采样和测试用例执行这 3 个主要步骤.IPT 首先通过不断地迭代划分输入域并确定性地选取划分后子域的中心点作为待执行的测试用例;随后基于执行策略,依次执行待选测试用例.当测试用例命中失效区域时,输出已执行的测试用例数量,否则将继续进入下一轮的迭代划分和中心点采样直至检测出程

序中的第 1 个失效，具体如算法 1 所示.

算法 1.　IPT.

输入：m 维待测程序的输入域 D，失效域 F

输出：检测到第 1 个失效时已执行的测试用例数量 F-count

1. 初始化 F-count=0，划分层次数 $n=0$

2. 选取 D 的中心点 a 执行，F-count++

IF 测试用例 a 命中失效区域

THEN RETURN F-count

ELSE n++

ENDIF

3(迭代划分). 根据已执行测试用例将当前输入域划分为 2^{mn} 个子域

4(中心采样). 采样 2^{mn} 个子域的中心点组成待测用例集 *TestSet*

5(用例执行). 当 *TestSet* 不为空时，根据执行策略，依次不放回地选取测试用例 $t\in$ *TestSet* 执行，F-count++

IF 测试用例 t 命中失效区域

THEN RETURN F-count

ENDIF

6. n++，转向步 3.

其中在步 4 获得 2^{mn} 个子域的中心点组成待测用例集后，步 5(用例执行)可以采用多种执行策略，包括基于等概率的执行策略，如随机执行、顺序执行等；以及基于不等概率的执行策略，如基于优先级的执行等.

对于给定的待测程序，在检测到第一个失效前，其失效率 θ 是确定不变的，但在测试过程中，θ 通常是未知的. 为此，IPT 采用了迭代划分、中心采样的策略，通过不断执行测试用例获取关于 θ 的信息. 在 m 维空间中，n 次划分并执行新增的 2^{mn} 个测试用例后，仍无法命中失效区域时，我们可以推断出该待测程序的失效率 θ 一定小于某个值. 如定理 1 所述.

定理 1.　假设输入域和失效域均为 m 维等距连续空间，失效率为 θ. 对于 IPT 有：若进行了 n 次划分，共计执行了 $\sum_{i=0}^{n}2^{m\cdot i}$ 个测试用例仍无法命中失效域，则可推断 θ 的取值范围为 $\theta<\frac{1}{2^{mn}}$.

证明.　可用数学归纳法证明，详见附录 1.

由定理 1 可知，在 m 维空间中，n 次划分并执行新增的 2^{mn} 个测试用例仍未命中失效区域时，可以推断出 $\theta<\frac{1}{2^{mn}}$. 换而言之，如果 $\theta=\frac{1}{2^{mn}}$，则最多需要 $\sum_{i=0}^{n}2^{m\cdot i}$ 个测试用例一定可以命中失效区域.

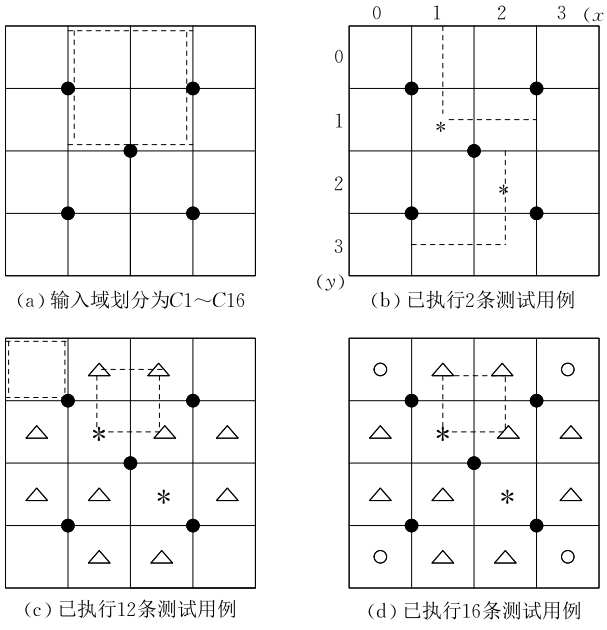
3.3　基于优先级的迭代划分测试方法(IPT-PS)

根据 IPT 的实现算法和定理 1，我们发现：只有待新增的 2^{mn} 个测试用例全部被执行通过后，才可以推断出 $\theta<\frac{1}{2^{mn}}$，否则仍然只能推断出 $\theta<\frac{1}{2^{m(n-1)}}$. 这促使我们考虑：$2^{mn}$ 个测试用例在执行过程中是否能获得更多关于 θ 的信息，不同的测试用例执行顺序是否会影响到 IPT 的失效检测能力.

基于上述考虑，我们提出了基于优先级的迭代划分测试方法 IPT-PS，通过分析新增的待执行的测试用例的不同空间特性，将测试用例分为 3 种不同优先等级后，依次执行，从而获取了更多关于 θ 的信息，可以得到当前待测程序的失效率 θ 的更为细致的推断.

我们通过一个例子说明 IPT-PS 如何实现基于优先级的用例执行，如图 2(a)所示：输入域和失效域均为二维等距连续空间，假定输入域的面积为 1. 前期已进行了 1 次划分，已执行的 5 个测试用例用符号“•”表示，2 次划分所产生 16 个子域的中心点组成待执行的测试用例集 $\{c_1\sim c_{16}\}$. 根据这些测试用例的空间特性，IPT-PS 将 16 个测试用例分为 3 种不同优先等级，对应集合为 T_1，T_2，T_3，在同一优先级的情况下，将随机执行其中任意一个测试用例.

测试用例优先级取决于该测试用例的执行是否会导致当前可放入的最大失效区域面积(记为 S)产生变化. 如图 2(a)～(d)所示，在加入 2 个“*”的测试用例前，$S=1/4$；当放入 2 个“*”的测试用例后，



(a) 输入域划分为C1～C16　(b) 已执行2条测试用例

(c) 已执行12条测试用例　(d) 已执行16条测试用例

图 2　基于优先级的测试用例执行示意图

S 将变更为 $\frac{1}{4}\times\frac{3}{4}\times\frac{3}{4}=\frac{9}{64}$；在加入 10 个“△”的测试用例后，$S$ 将变更为 $\frac{1}{4}\times\frac{1}{4}=\frac{1}{16}$；在加入最后 4 个“○”的测试用例后，$S$ 保持不变. 由此可见，在图 2 中，标记为“＊”的测试用例属于集合 T_1，优先级最高；标记为“△”的测试用例属于集合 T_2，优先级次之；标记为“○”的测试用例属于集合 T_3，优先级最低.

可放入的最大失效区域面积 S 的不断缩小，意味着随着测试用例的执行通过，可以推断出的该待测程序的失效率 θ 的取值范围也在不断缩小，我们可以获得更加精确的关于 θ 可能取值的推断.

为实现 IPT-PS，我们首先需要知道如何将待执行测试用例分类，并分别放入 T_1，T_2，T_3 集合中. 如图 2(b)所示，我们为每一维经过 n 次划分后得到的 2^n 个子域的中心点，即 2^n 个待执行测试用例建立索引，记为 $I_k=\{0,1,\cdots,2^n-1\}$，其中下标 k 表示第 k 维. 这样，在二维空间中，待执行的 2^{2n} 个测试用例可以用 (i_1,i_2) 唯一标记，其中 $i_1\in I_1$，$i_2\in I_2$. 观察图 2 可以发现，T_1 中的元素需同时满足以下两个条件：(1) 每一维上的索引不为 0 或最大值(即 $i_1\neq 0$，2^n-1 且 $i_2\neq 0,2^n-1$)；(2) $|i_1-i_2| \bmod 2$ 为 0；T_3 中的元素满足：每一维上的索引为 0 或最大值；T_2 则为 2^{2n} 个待测用例中不属于 T_1 和 T_3 的测试用例构成的集合.

当从二维空间推广到 m 维空间时，经过 n 次划分得到的 2^{mn} 个待测用例可以用 $(i_1,i_2,\cdots,i_m)$ 唯一标记，其中 $i_k\in I_k$. 此时，可以将 2^{mn} 个待测用例按照如下规则分别放入 T_1，T_2，T_3.

T_1 中的元素满足：(1) 每一维上的索引不为 0 或最大值(即 $i_k\neq 0,2^n-1$)；(2) $|i_j-i_{j+1}| \bmod 2$ 为 0，其中 $j\in\{1,2,\cdots,m-1\}$.

T_3 中的元素满足：每一维上的索引为 0 或最大值.

T_2 为 2^{mn} 个待测用例中不属于 T_1 和 T_3 的测试用例构成的集合.

基于上述规则，可分别计算集合 T_1，T_2，T_3 的大小. 对于 T_1 而言：由条件(1)可知，每一维上的索引 $i_k\in I_k$ 只能从 $\{1,2,\cdots,2^n-2\}$ 中取值，因此当前待选测试用例总个数为 $(2^n-2)^m$. 由条件(2)可知，测试用例需要满足 $m-1$ 次 $|i_j-i_{j+1}| \bmod 2$ 为 0 的判断，而 $|i_j-i_{j+1}| \bmod 2$ 取值为 0 或 1 的测试用例数量是相同的，因此满足条件(2)的测试用例数量只占满足条件(1)的测试用例数量的 $\frac{1}{2^{m-1}}$，即 T_1 中的元素个数为 $\frac{(2^n-2)^m}{2^{m-1}}$. 对 T_3 而言，每一维上的索引 $i_k\in I_k$ 只能从 $\{0,2^n-1\}$ 中取值，因此，T_3 中的元素个数为 2^m. 而 T_2 中的元素个数则为 $2^{mn}-\frac{(2^n-2)^m}{2^{m-1}}-2^m$.

在已知每个测试用例所从属的集合后，IPT-PS 仅需在 IPS 的算法框架上将步 5(用例执行)调整为基于优先级的用例执行，算法描述如算法 2 所示.

算法 2. IPT-PS.

输入：m 维待测程序的输入域 D，失效域 F

输出：检测到第 1 个失效时已执行的测试用例数量 F-count

步 1～步 4：同算法 1(IPT)

5(基于优先级的用例执行). 根据优先级高低，将 2^{mn} 个待测用例分别放入集合 T_1，T_2，T_3

依次针对集合 $T_i(i=1,2,3)$

WHILE ($T_i\neq\varnothing$)

随机从 T_i 选择测试用例 $t_j(j=1,2,\cdots,|T_i|)$ 执行

$T_i=T_i-\{t_j\}$，F-count++

IF 测试用例 t_j 命中失效区域

THEN RETURN F-count

ENDIF

ENDWHILE

6. n++，转向步 3.

根据 IPT-PS 基于优先级执行测试用例的特点，我们发现在待执行的共计 2^{mn} 个测试用例中，当执行了优先级最高的 T_1 中的测试用例无法命中失效区域时，可以推断出该待测程序的失效率 θ 一定小于某个值，如定理 2 所述.

定理 2. 假设输入域和失效域均为 m 维等距连续空间，失效率为 θ. 对于 IPT-PS 有：若进行了 n 次划分，共计执行了 $\sum_{i=0}^{n-1}2^{m\cdot i}+\frac{(2^n-2)^m}{2^{m-1}}$ 个测试用例仍无法命中失效域，则可推断 θ 的取值范围为 $\theta<\frac{1}{2^{m(n-1)}}\left(\frac{3}{4}\right)^m$.

证明. 可用数学归纳法证明，详见附录 1.

由定理 1 和定理 2 可知，在 m 维空间中，进行 n 次划分后，IPT 需完全执行 2^{mn} 个测试用例，才可推断 θ 的取值范围将从 $\theta<\frac{1}{2^{m(n-1)}}$ 缩小至 $\theta<\frac{1}{2^{mn}}$，而 IPT-PS 引入基于优先级执行测试用例，在执行了 $\frac{(2^n-2)^m}{2^{m-1}}$ 个测试用例后，即可推断 θ 的取值范围将从 $\theta<\frac{1}{2^{m(n-1)}}$ 缩小至 $\theta<\frac{1}{2^{m(n-1)}}\left(\frac{3}{4}\right)^m$.

此外，IPT-PS 中 T_1 的元素个数为 $\frac{(2^n-2)^m}{2^{m-1}}$，因为 $\frac{(2^n-2)^m}{2^{m-1}}<\frac{2^{mn}}{2^{m-1}}$，意味着 IPT-PS 仅需使用少于本轮待执行测试用例总量 2^{mn} 的 $\frac{1}{2^{m-1}}$ 个测试用例，即可获得关于 θ 的取值范围的更细致的推断.

3.4　IPT 和 IPT-PS 的理论分析

由定理 1 和定理 2 可知，当待测程序的 $\theta=\frac{1}{2^{mn}}$ 时，IPT 最多需要执行 $\sum_{i=0}^{n}2^{m\cdot i}$ 个测试用例就可以命中失效区域，而 IPT-PS 最多在执行完第 n 次划分的新增测试用例中 T_1，T_2 的测试用例后，就可以命中失效区域. 当 $\theta=\frac{1}{2^{m\cdot(n-1)}}\left(\frac{3}{4}\right)^m$ 时，IPT 仍然最多需要执行 $\sum_{i=0}^{n}2^{m\cdot i}$ 个测试用例才可以命中失效区域，而 IPT-PS 最多在执行完第 n 次划分的新增测试用例中 T_1 的测试用例后，就可以命中失效区域. 因此，借助 θ 进一步对比 IPT 和 IPT-PS 在最坏情况下的表现，有如下分析：

(1) 对于 IPT，令 $N_{\text{IPT}}^{\text{worst}}$ 为最坏情况下 IPT 所需执行的测试用例数量.

当 $\frac{1}{2^{m\cdot n}}\le\theta<\frac{1}{2^{m\cdot(n-1)}}$ 时，有 $2^{m(n+1)}<\frac{2^{2m}}{\theta}$，

所以 $N_{\text{IPT}}^{\text{worst}}\le\frac{2^{m\cdot(n+1)}-1}{2^m-1}<\frac{4^m}{2^m-1}\cdot\frac{1}{\theta}-\frac{1}{2^m-1}$.

(2) 对于 IPT-PS，令 $N_{\text{IPT-PS}}^{\text{worst}}$ 为最坏情况下 IPT-PS 所需执行的测试用例数量.

① 当 $\frac{1}{2^{m\cdot n}}\le\theta<\frac{1}{2^{m\cdot(n-1)}}\cdot\left(\frac{3}{4}\right)^m$ 时，有 $2^{m(n+1)}<\frac{3^m}{\theta}$，所以

$$N_{\text{IPT-PS}}^{\text{worst}}\le\frac{2^{m\cdot(n+1)}-1}{2^m-1}-2^m<\frac{2^{m\cdot(n+1)}-1}{2^m-1}<\frac{\frac{3^m}{\theta}-1}{2^m-1}=\frac{3^m}{2^m-1}\cdot\frac{1}{\theta}-\frac{1}{2^m-1}.$$

② 当 $\frac{1}{2^{m\cdot(n-1)}}\cdot\left(\frac{3}{4}\right)^m\le\theta<\frac{1}{2^{m\cdot(n-1)}}$ 时，有 $2^{mn}<\frac{2^m}{\theta}$，所以

$$N_{\text{IPT-PS}}^{\text{worst}}\le\frac{2^{m\cdot n}-1}{2^m-1}+\frac{(2^n-2)^m}{2^{m-1}}<\frac{2^{m\cdot n}}{2^m-1}+\frac{2^{m\cdot n}}{2^{m-1}}<\frac{3}{2^m-1}\cdot2^{m\cdot n}<\frac{3\cdot2^m}{2^m-1}\cdot\frac{1}{\theta}.$$

令 Δ 为上述①和②两种情况下 $N_{\text{IPT-PS}}^{\text{worst}}$ 的差值，则

$$\Delta=\frac{3^m}{2^m-1}\cdot\frac{1}{\theta}-\frac{1}{2^m-1}-\frac{3\cdot2^m}{2^m-1}\cdot\frac{1}{\theta}=\frac{3^m-3\cdot2^m}{2^m-1}\frac{1}{\theta}-\frac{1}{2^m-1}\ge\frac{3^m-3\cdot2^m}{2^m-1}-\frac{1}{2^m-1}.$$

当 $m\ge3$ 时，$\Delta>0$ 成立. 综上可得

$$N_{\text{IPT-PS}}^{\text{worst}}=\begin{cases}\frac{3\cdot2^m}{2^m-1}\cdot\frac{1}{\theta}, & m=1,2\\ \frac{3^m}{2^m-1}\cdot\frac{1}{\theta}-\frac{1}{2^m-1}, & m\ge3\end{cases}.$$

由上述分析可知，已知待测程序的 θ，IPT 和 IPT-PS 均存在着检测出对应失效所需测试用例数量的上界. 当 $m\ge2$ 时，$N_{\text{IPT-PS}}^{\text{worst}}<N_{\text{IPT}}^{\text{worst}}$ 成立，因此 IPT-PS 获得更紧的上界. 相对于随机测试的不确定性，IPT 和 IPT-PS 存在上界将为测试过程中测试资源的有效分配提供帮助.

4　实验结果与分析

本节将给出一系列实验结果，并围绕以下 3 个研究问题展开分析和对比：

问题 1. IPT-PS 在失效检测能力和时间开销方面的表现如何？

问题 2. 相对于 IPT，IPT-PS 采用基于优先级的用例执行策略是否能有效地改进算法性能？

问题 3. IPT-PS 在高维、多种失效模式、真实程序等场景下的适用性如何？

4.1　实验设置

本文首先采用仿真实验进行比较分析，仿真实验将有利于帮助控制实验参数的变化，涉及的参数主要包括维度、失效率、测试方法和实验次数，逐一说明如下：

(1) 维度. 当待测程序有 m 个输入参数时，该程序的输入域为 m 维空间. 在仿真实验中，将讨论 1～4 维空间输入域，并假定均为各维等距的连续空间，即 2 维输入域为正方形，3 维输入域为正方体.

(2) 失效率. 即失效输入域占整个输入域的比例，显然失效率 $\theta\in[0,1]$. 同时假定，失效输入域也是各维等距的连续空间.

(3) 测试方法. 本次实验中将对比以下 6 种测试方法：

①RT. 有放回的随机测试将作为本实验的基准方法.

②FSCS. 根据文献[24]的结果，FSCS-ART 的失效检测能力较强，可作为基于距离计算的 ART

算法的代表.其中,固定候选集大小为 10.

③ IP-ART.根据文献[24]的结果,IP-ART 的计算开销较小,可作为基于划分的 ART 算法的代表.其中,划分参数 p 初始化为 1.

④ PSS.当仅已知输入域的空间信息时,PSS 仍然可用,因此选取 PSS 作为传统划分测试的代表.PSS 要求从每个子域中有放回地随机选取的测试用例数应与该子域的大小成固定比例.本次实验中,为便于比较,我们在 IPT 迭代划分所产生的每个相同大小的子域中随机选取一个测试用例,而非固定地选取子域的中心点.

⑤ IPT.本文提出的迭代划分、中心采样的划分测试方法.

⑥ IPT-PS.本文提出的迭代划分、中心采样和基于优先级的用例执行的划分测试方法.

(4)实验次数.重复一定的实验次数是为了有效避免随机性对实验结果的影响.在本文实验中,涉及两个方面的随机性:一是失效域位置的随机性,二是测试用例选择的随机性.这里,我们重复 1000 次实验来克服失效域位置的随机性.在给定失效域位置的前提下,通过重复 2000 次实验来克服测试用例选择的随机性.

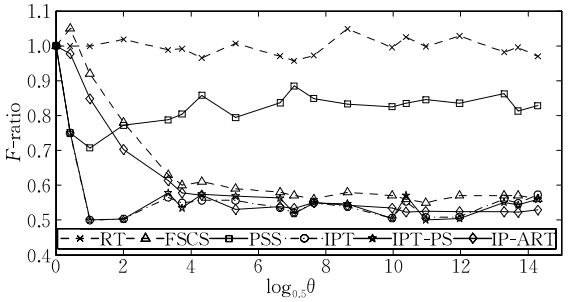
给定上述实验参数配置,实验过程如下:首先根据维度产生输入域,并根据失效率计算并产生失效域.在每一次实验中,随机放置失效域,根据测试方法逐个选择测试用例.当测试用例落入失效域中则认为其检测到失效,记录当前已执行的测试用例数量 F-count.当完成规定实验次数时,F-count 的平均值记为 F-measure,并记录实验所耗费的总时间.

本文实验的度量包括有效性度量和效率度量.在有效性度量方面,我们采用将各种方法的 F-measure 与 RT 的 F-measure 的理论值(即 $1/\theta$)的比值记为 F-ratio 来衡量各种方法在失效检测能力上相对 RT 的改进.显然,F-ratio 越小,则该方法相对 RT 的改进越明显.在效率度量方面,我们采用各种方法完成一次失效检测所需的平均时间,记为 *Runtime* 来比较各种方法的计算开销.

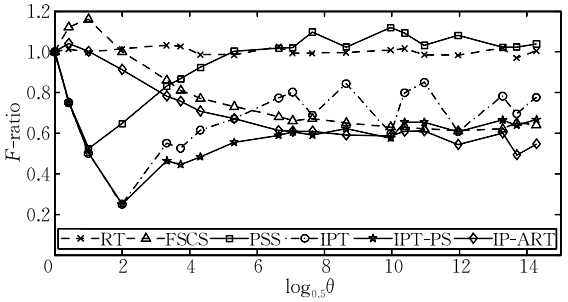
4.2 问题 1:相关方法的分析与比较

我们通过实验 1 来比较 RT,FSCS,IP-ART,PSS,IPT 和 IPT-PS 这 6 种测试方法在 F-ratio 和 *Runtime* 这两个度量上的表现.实验中参数维度为 1～4 维,失效率 $\theta=\{1, 0.75, 0.5, 0.25, 0.1, 0.075, 0.05, 0.025, 0.01, 0.0075, 0.005, 0.0025, 0.001, 0.00075, 0.0005, 0.00025, 0.0001, 0.000075, 0.00005\}$,维度和失效率的选取与文献[32]相同.

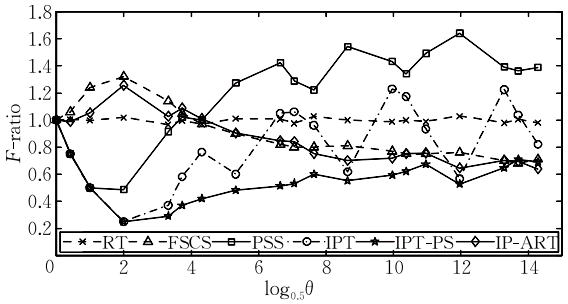
如图 3 所示,图 3(a)～(d)分别为 1 维到 4 维输入域下,6 种测试方法的 F-ratio 比较,其中 x 轴为 $\log_{0.5}\theta$ 以便于数据的展示,y 轴为 F-ratio.类似地,如图 4 所示,图 4(a)～(d)分别为 1 维到 4 维输入域下,6 种测试方法的 *Runtime* 比较,其中 x 轴为 $\log_{0.5}\theta$,y 轴为 *Runtime*,单位为 ms.



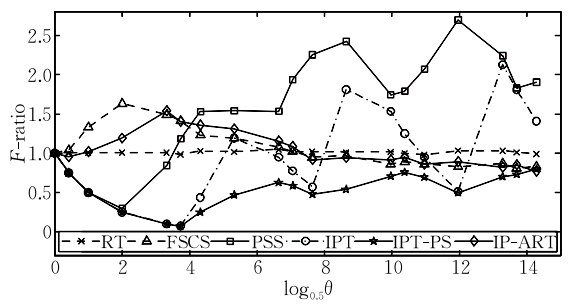
(a) 1维输入域F-ratio对比



(b) 2维输入域F-ratio对比



(c) 3维输入域F-ratio对比

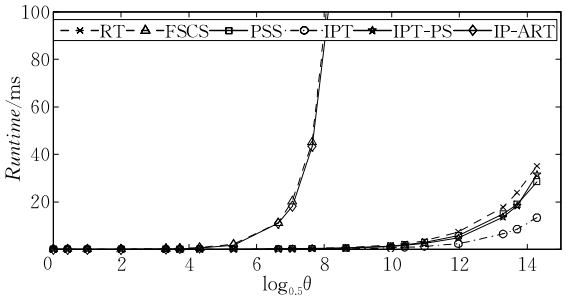


(d) 4维输入域F-ratio对比

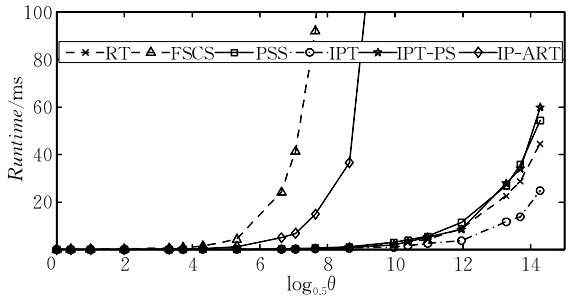
图 3 6 种测试方法的 F-ratio 比较

以 RT 为基准方法，由图 3 可知：(1) 对于 FSCS 和 IP-ART，两者在 *F*-ratio 方面的表现类似，在 4 个维度下均有：当 θ 较大时，*F*-ratio 大于 1，随着 θ 的减小，*F*-ratio 逐渐降低、趋于稳定且小于 1. 在给定 θ 下，维度越高，对应的 *F*-ratio 越大；(2) 对于 PSS，IPT 和 IPT-PS，在 4 个维度下均有：*F*-ratio 在下降至最低点后，随着 θ 的减小，*F*-ratio 逐渐增大并呈周期性波动. 此外，随着维度的增加，*F*-ratio 越大且波动幅度越大，在 3 维和 4 维输入域下，PSS 和 IPT 的部分 *F*-ratio 将大于 1. 3 种方法的不同点在于，在给定 θ 的情况下，PSS 的 *F*-ratio 最大，而 IPT-PS 的 *F*-ratio 最小.

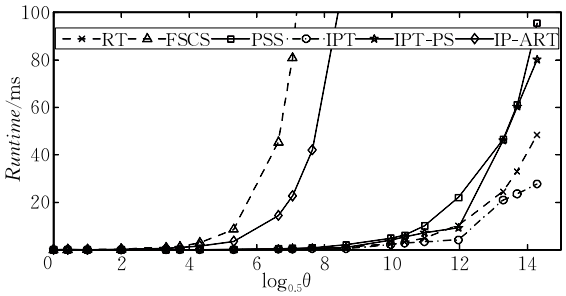
以 RT 为基准方法，由图 4 可知：(1) 对于 FSCS 和 IP-ART，随着 θ 的减小，*Runtime* 快速上升. 较之 FSCS，尽管在 2～4 维输入域下，IP-ART 的 *Runtime* 有明显减少，但仍然远远大于 RT 的 *Runtime*. 此外，在 1 维输入域下，由于 IP-ART 需频繁地划分输入域，耗费了大量的计算开销，其 *Runtime* 与 FSCS 的 *Runtime* 很接近；(2) PSS，IPT 和 IPT-PS 三者的 *Runtime* 与 RT 的 *Runtime* 相当，随着维度的增加、θ 的减小，三者的 *Runtime* 将略大于 RT 的 *Runtime*.



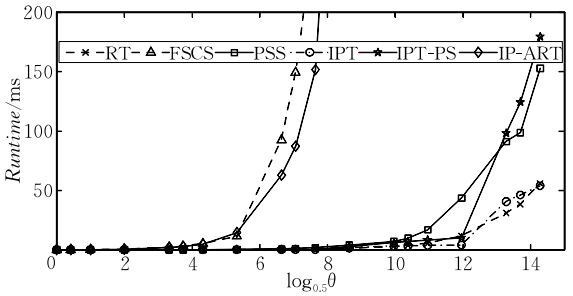
(a) 1维输入域*Runtime*对比



(b) 2维输入域*Runtime*对比



(c) 3维输入域*Runtime*对比



(d) 4维输入域*Runtime*对比

图 4　6 种测试方法的 *Runtime* 比较

下面我们将 IPT-PS 与 FSCS、IP-ART、PSS、IPT 逐一分析比较，并讨论相应的原因.

(1) IPT-PS 与 ART(FSCS、IP-ART)的讨论

在 *F*-ratio 方面，当 θ 较大时，IPT-PS 仅需要通过很少次数的划分就可以检测到失效，*F*-ratio 明显优于 ART(以 FSCS、IP-ART 为例). 当 θ 较小时，两者的 *F*-ratio 相近，ART 的 *F*-ratio 较稳定，IPT-PS 则存在着波动.

在 *Runtime* 方面，当 θ 较大时，两者差别不大. 然而，随着维度的提高和 θ 的减小，ART 的 *Runtime* 将远远高于 IPT-PS 的 *Runtime*. 如表 1 所示，当 θ 为 0.01 时，IPT-PS 的 *Runtime* 通常是 ART 的 *Runtime* 的 1%；当 θ 降低至 0.0001 时，IPT-PS 在 1～4 维输入域上的时间性能比 ART 提高了 3 个数量级.

表 1　*Runtime* 对比　(单位：ms)

失效率	测试方法	维度			
		1D	2D	3D	4D
0.01	RT	0.18	0.23	0.26	0.30
	IPT-PS	0.16	0.26	0.36	0.42
	IP-ART	11.04	4.97	14.42	62.72
	FSCS	11.18	24.12	45.29	92.52
0.0001	RT	17.75	22.54	24.41	31.08
	IPT-PS	13.6	27.8	46.1	98.3
	IP-ART	9.10×10^4	1.51×10^4	5.44×10^4	1.70×10^5
	FSCS	8.62×10^4	1.48×10^5	3.80×10^5	4.62×10^5

由此可见，随着 θ 的减小，尽管 ART 能够改进 RT 的失效检测能力，但是其时间开销远远大于 RT. IPT-PS 借鉴了 ART 均匀分布的思想，通过迭代划分、中心采样和基于优先级的用例执行这 3 个操作实现了输入域上的均匀分布，然而这 3 项操作均无需耗费过多的计算开销，其时间开销保持在与

RT 同一数量级.

此外,当 θ 较小时,IPT-PS 存在着进一步改进的可能. 在没有任何 θ 的先验知识的情况下,IPT-PS 从第 1 轮划分开始逐渐增加对 θ 的认知,不断推断出 θ 的取值范围的变化. 当存在 θ 的先验知识,特别是当可预测出 θ 的取值很小,例如当预测出 $\theta<0.0001$ 时,IPT-PS 可以通过相应计算后直接进行第 n 次划分,从而节省了前面 $n-1$ 次划分所耗费的中心点测试用例,从而改进了当 θ 较小时的 *F*-ratio 及 *Runtime*.

(2) IPT-PS 与 PSS 的讨论

与 PSS 相比,IPT-PS 通过确定性的中心点采样以及基于优先级的用例执行,可以获得较优的 *F*-ratio. 这是因为,在迭代划分子域里中心点采样比子域里随机采样更能确保测试用例在输入域上的均匀分布,进而改进失效检测能力. 在计算开销方面,两者 *Runtime* 相当.

(3) IPT-PS 与 IPT 的讨论

与 IPT 相比,IPT-PS 引入基于优先级的用例执行,可以获得较优的 *F*-ratio. 这是因为,IPT-PS 通过指定测试用例的执行序列,进一步在两轮迭代上通过优先级采样实现了测试用例更均匀的分布. 在计算开销方面,IPT-PS 由于引入了优先级相关的开销,*Runtime* 略大于 IPT.

为进一步验证 IPT-PS 能够显著改进 RT 的失效检测能力,获得与 ART 相近的失效检测能力,我们对 IPT-PS 与其他 5 个对比算法即 RT,FSCS,IP-ART,PSS,IPT 的 *F*-ratio 展开了 *T*-检验,其中,显著性水平值为 0.05. 双边 *T*-检验中,假设 H_0:*A* 与 *B* 不存在显著差异;对应有假设 H_1:*A* 与 *B* 存在显著差异. 当假设 H_1 成立时,我们将进一步开展单边 *T*-检验,假设 H_0':*A* 不显著地大于 *B*;对应有假设 H_1':*A* 显著地大于 *B*.

由于实验数据较多,这里我们仅呈现了部分假设检验的结果. 以 2 维输入域下,等距选取实验 1 中 θ 的部分取值为例,即 $\theta=\{0.05,0.0075,0.001,0.00025,0.00005\}$,具体检验数据如表 2 所示:当 $\theta=0.05$ 时,假设 H_1' 均成立,即 5 种对比算法的 *F*-ratio 均显著地大于 IPT-PS 的 *F*-ratio. 当 $\theta=\{0.0075,0.00005\}$ 时,RT,PSS,IPT 对应假设 H_1' 成立,即 RT,PSS,IPT 的 *F*-ratio 均显著地大于 IPT-PS 的 *F*-ratio;FSCS,IP-ART 对应假设 H_0 成立,即 FSCS,IP-ART 的 *F*-ratio 与 IPT-PS 的 *F*-ratio 不存在显著差异. 当 $\theta=\{0.001,0.00025\}$ 时,RT,PSS 对应假设 H_1' 成立,即 RT,PSS 的 *F*-ratio均显著地大于 IPT-PS 的 *F*-ratio;FSCS,IP-ART,IPT 对应假设 H_0 成立,即 FSCS,IP-ART,IPT 的 *F*-ratio 与 IPT-PS 的 *F*-ratio 不存在显著差异.

结合实验 1 的数据和假设检验的结果,我们可以发现:IPT-PS 能够显著改进 RT 的失效检测能力,并获得与 ART 相近甚至更好的失效检测能力.

表 2 IPT-PS 与对比算法关于 *F*-ratio 的 *T*-检验数据

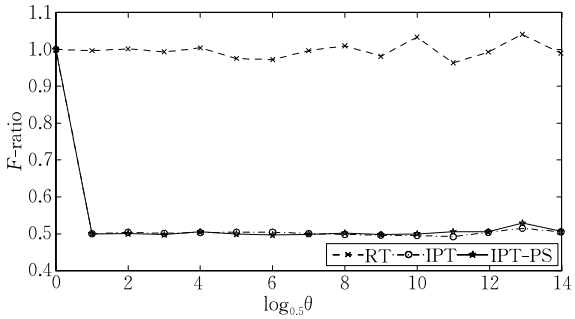
θ	*A*	*B*	2-tailed accepted hypothesis (*p*-value)	1-tailed accepted hypothesis (*p*-value)
0.05	RT	IPT-PS	H_1: 2.38416E-29	H_1': 1.19208E-29
	FSCS		H_1: 2.67088E-15	H_1': 1.33544E-15
	IP-ART		H_1: 2.94734E-4	H_1': 1.47367E-4
	PSS		H_1: 1.05401E-20	H_1': 5.27006E-21
	IPT		H_1: 0.00726	H_1': 0.00363
0.0075	RT	IPT-PS	H_1: 9.20997E-15	H_1': 4.60498E-15
	FSCS		H_0: 0.07456	—
	IP-ART		H_0: 0.72451	—
	PSS		H_1: 8.85406E-12	H_1': 4.42703E-12
	IPT		H_1: 0.03728	H_1': 0.01392
0.001	RT	IPT-PS	H_1: 6.68685E-18	H_1': 3.34343E-18
	FSCS		H_0: 0.35053	—
	IP-ART		H_0: 0.82403	—
	PSS		H_1: 4.41601E-12	H_1': 2.208E-12
	IPT		H_0: 0.22024	—
0.00025	RT	IPT-PS	H_1: 1.04301E-13	H_1': 5.21503E-14
	FSCS		H_0: 0.71539	—
	IP-ART		H_0: 0.44215	—
	PSS		H_1: 1.72008E-4	H_1': 8.60038E-5
	IPT		H_0: 0.75189	—
0.00005	RT	IPT-PS	H_1: 1.63438E-6	H_1': 8.17189E-7
	FSCS		H_0: 0.68743	—
	IP-ART		H_0: 0.79075	—
	PSS		H_1: 5.50093E-5	H_1': 2.75046E-5
	IPT		H_1: 0.01988	H_1': 0.00994

综合上述比较分析,我们可以回答问题 1:IPT-PS 在仅使用接近 RT 时间开销的情况下,可以达到与 ART 相近甚至更好的失效检测能力,在有效性和效率之间实现了良好的平衡.

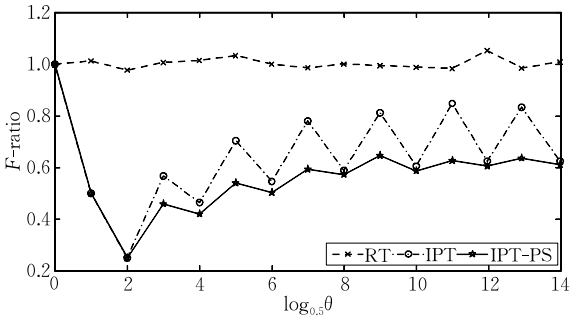
4.3 问题 2:IPT 和 IPT-PS 的分析与比较

基于实验 1 中关于 IPT 与 IPT-PS 的初步结果,我们通过实验 2 来进一步分析 IPT 与 IPT-PS 的特性.

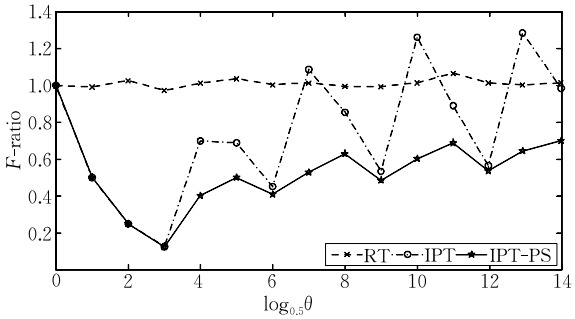
在实验 2 中,维度为 1～4 维,失效率 $\theta=\{1,2^{-1},2^{-2},2^{-3},2^{-4},2^{-5},2^{-6},2^{-7},2^{-8},2^{-9},2^{-10},2^{-11},2^{-12},2^{-13},2^{-14}\}$. 选取 θ 的值为 2^{-n},是希望通过实验验证 IPT 与 IPT-PS 中关于执行测试用例数与可推断的 θ 取值范围的关系. 如图 5 所示,图 5(a)～(d)分别为 1 维到 4 维输入域下,IPT 与 IPT-PS 的 *F*-ratio 比较,其中 x 轴为 $\log_{0.5}\theta$,y 轴为 *F*-ratio. 这



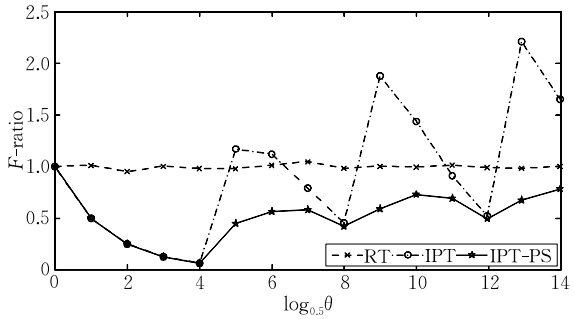
(a) 1维输入域F-ratio对比



(b) 2维输入域F-ratio对比



(c) 3维输入域F-ratio对比



(d) 4维输入域F-ratio对比

图 5　$\theta=2^{-n}$时 IPT 与 IPT-PS 的 F-ratio 比较

里，由于 θ 的取值为 2^{-n}，因此获得的数据点在 x 轴上是等距的.

以 RT 作为基准方法，由图 5 除了可以获得与图 3 类似的结论外，还可以获得如下结论：

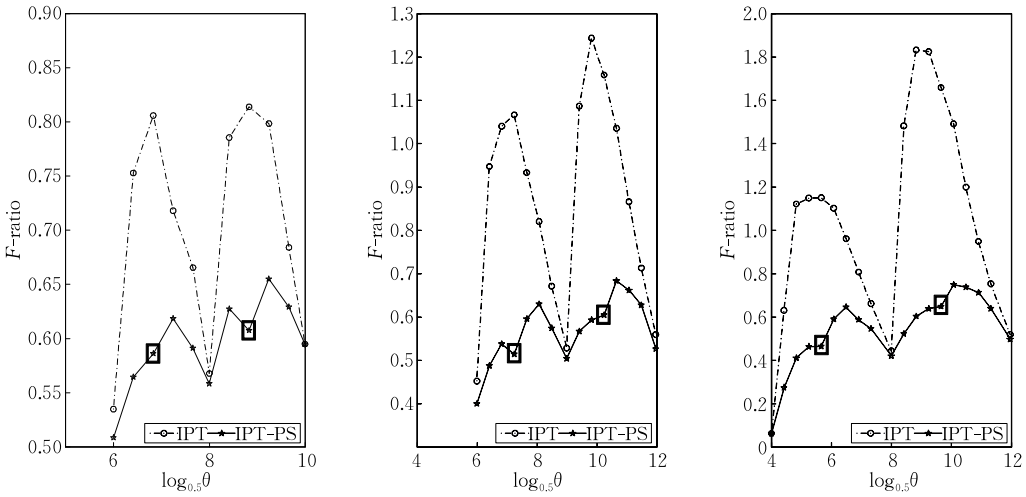
(1) IPT 和 IPT-PS 在 4 个维度下均存在 F-ratio 的最小值. 以 2 维输入域为例，当 $\theta=2^{-2}$，$\log_{0.5}\theta=2$ 时，F-ratio 取到最小值，为 0.25. 这是因为，IPT 和 IPT-PS 均仅需要执行 1 条测试用例，即取到当前 2 维输入域的中心点时即可检测到失效. 同理，在 m 维输入域下，当 $\theta=2^{-m}$，$\log_{0.5}\theta=m$ 时，仅执行 1 条测试用例即可检测到失效，此时 F-ratio 取到最小值，为 2^{-m}.

(2) 在 4 个维度下均有：每隔固定间隔，存在 IPT 和 IPT-PS 的 F-ratio 非常接近的汇集点. 以 2 维输入域为例，当 $\log_{0.5}\theta=\{2,4,6,8,10,12,14\}$ 时，两种方法的 F-ratio 非常接近. 这是因为在 2 维输入域中，当 $\theta=2^{-2n}$ 时，IPT 在最坏情况下所需的测试用例数为 $1+2^2+\cdots+2^{2n}$ 个，而 IPT-PS 基于优先级将待测用例分为 T_1，T_2，T_3 依次执行，在最坏情况下所需的测试用例数为 $|T_1|+|T_2|$，即 $1+2^2+\cdots+2^{2n}-2^2$ 个. 因此，当实验重复足够次数时，IPT 和 IPT-PS 的 F-ratio 将非常接近. 同理，在 m 维输入域下，当 $\theta=2^{-mn}$，$\log_{0.5}\theta=mn$ 时，IPT 与 IPT-PS 的 F-ratio 非常接近.

我们从实验 1 和实验 2 中均发现，相对于 IPT，IPT-PS 的 F-ratio 较小，且波动幅度较小. 为了进一步分析 IPT-PS 采用基于优先级的用例执行策略对 F-ratio 的影响，我们开展了实验 3.

在实验 3 中，我们选取 2～4 维输入域下的特定的 θ 进行更为细化的 IPT 与 IPT-PS 的 F-ratio 比较. 2 维输入域下，$\theta=\{2^{-6},2^{-6}\times(3/4)^i,2^{-8},2^{-8}\times(3/4)^i,2^{-10}\}$，$i=1,2,3,4$，共计 11 个取值；3 维输入域下，$\theta=\{2^{-6},2^{-6}\times(3/4)^i,2^{-9},2^{-9}\times(3/4)^i,2^{-12}\}$，$i=1,2,\cdots,6$，共计 15 个取值；4 维输入域下，$\theta=\{2^{-4},2^{-4}\times(3/4)^i,2^{-8},2^{-8}\times(3/4)^i,2^{-12}\}$，$i=1,2,\cdots,8$，共计 19 个取值.

实验结果如图 6(a)～(c)所示，当 θ 取到某些特定值时，IPT-PS 的 F-ratio 较之 IPT 有明显的下降，从而使得波动幅度变小. 我们用矩形框在图 6 中标注出这些特殊的数据点，通过分析这些数据点的 θ 取值发现：2 维输入域下，$\theta=2^{-6}\times(3/4)^2$，$2^{-8}\times(3/4)^2$；3 维输入域维下，$\theta=2^{-6}\times(3/4)^3$，$2^{-9}\times(3/4)^3$；4 维输入域下，$\theta=2^{-4}\times(3/4)^4$，$2^{-8}\times(3/4)^4$. 这与我们在前文讨论的理论结果一致，即当 θ 取值为 $\frac{1}{2^{m(n-1)}}\left(\frac{3}{4}\right)^m$（$m$ 为维度，n 为划分次数）时，IPT-PS 在最坏情况下仅需执行优先级最高的 $|T_1|$ 个测试用例，即可检测到失效；而 IPT 在最坏情况下则需执行所有新增测试用例，即 $|T_1|+|T_2|+|T_3|$ 个测试用例才可检测到失效. 因此，引入了基



(a) 2维输入域F-ratio对比 (b) 3维输入域F-ratio对比 (c) 4维输入域F-ratio对比

图 6 细化的 IPT 与 IPT-PS 的 F-ratio 比较

于优先级的用例执行策略后,IPT-PS 的 F-ratio 较之 IPT 有明显的下降.

因此,我们可以回答问题 2:较之 IPT,IPT-PS 采用基于优先级的用例执行策略能够有效改进算法的 F-ratio.

4.4 问题 3:IPT-PS 的适用性分析

鉴于前期的实验是面向 1～4 维输入域的块状失效模式下的实验,本节将初步探究 IPT-PS 的适用性,主要包括高维、多种失效模式、真实程序场景下 IPT-PS 的适用性.

(1)维度的扩展

针对 5～8 维输入域的块状失效模式,失效率 $\theta=2^{-n}(n=0,1,\cdots,14)$,获取 IPT-PS 的 F-ratio 如图 7 所示. 与前期实验观察一致,IPT-PS 的 F-ratio 在下降至对应的最低点后,随着 θ 的减小,F-ratio 逐渐增大并呈周期性波动. 此外,随着维度的增加,相应的 F-ratio 越大且波动幅度越大. 在 7 维和 8 维输入域下,存在部分 F-ratio 大于 1.

由此可见,IPT-PS 在高维场景下仍然可用,当 θ 较大时,IPT-PS 仍然保持较高的失效检测能力. 当 θ 较小时,F-ratio 存在着波动. 因此,如何改进 IPT-PS 在高维的性能,减小 IPT-PS 的波动幅度,将是我们后续的一项重要工作.

(2)失效模式的扩展

为验证 IPT-PS 在多种失效模式下的适用性,我们将 IPT-PS 分别应用于圆形失效模式和点状失效模式[32].

在圆形失效模式下,维度为 1～4 维,失效率 $\theta=2^{-n}(n=0,1,\cdots,14)$,得到 IPT-PS 的 F-ratio 如图 8 所示. 与 IPT-PS 在块状失效模式下的表现(参见 4.3 节图 5)类似,F-ratio 在下降至对应的最低点后,随着 θ 的减小,F-ratio 逐渐增大并呈周期性波动. F-ratio 始终小于 1,说明 IPT-PS 在圆形失效模式下仍然适用,仍具备较强的失效检测能力.

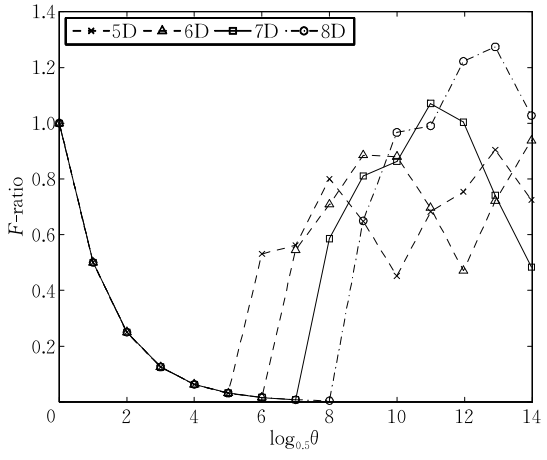


图 7 5～8 维输入域下 IPT-PS 的 F-ratio

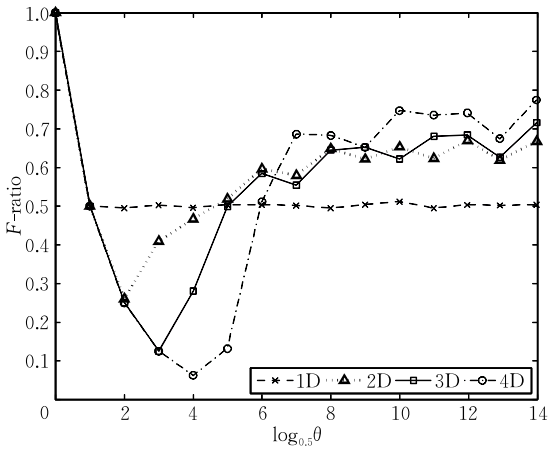


图 8 圆形失效模式下 IPT-PS 的 F-ratio

文献[32]中指出，点状失效模式的核心不在于失效区域的大小，而在于失效区域的数量. 因此，本文参考了文献[32]中的实验参数，选取维度为 2 维，失效率 $\theta=\{0.005, 0.001, 0.0005\}$，等大小的方形失效区域的数量 $n=\{1, 4, 7, 10, 20, 30, \cdots, 90, 100\}$，获得点状失效模式下 IPT-PS 的 F-ratio 如图 9 所示. 由图中可以获得两个结论：(1) 在相同的 θ 取值下，随着 n 的增加，F-ratio 将逐渐增大. 这是因为，n 的增加将直接导致单个失效区域对应的实际 θ 减小，从而使得 F-ratio 逐渐增大；(2) 在相同的 n 取值下，θ 的取值越小，F-ratio 越大，这一观察与前期实验结果一致.

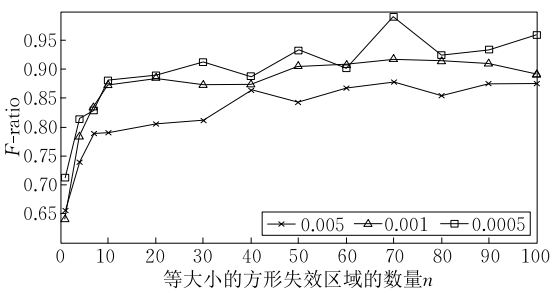


图 9　点状失效模式下 IPT-PS 的 F-ratio

上述两个实验说明，IPT-PS 在圆形和点状失效模式下仍适用，且具备较强的失效检测能力.

(3) 真实程序的验证

我们从 Numerical Recipes 中选择了两个 C 程序 bessj0 和 select[33] 来分析 IPT-PS 的适用性. bessj0 的输入为一个浮点数(float x)，输出为一个浮点数. select 的输入为两个整数和一个浮点数数组(unsigned long k, unsigned long n, float $arr[]$)，输出为一个浮点数，即 n 维数组中第 k 小的数. 我们采用 Csaw 工具随机生成变异[34]，通过选择、简化，最终 bessj0 共有 140 个有效变异，select 共有 8 个有效变异. 针对每个有效变异，重复 30 次实验并分别记录 RT 和 IPT-PS 为检测出该变异所需测试用例的数量. 令 N_{RT}，$N_{IPT\text{-}PS}$ 分别为 30 次实验下 RT 和 IPT-PS 所需测试用例数量的均值，令 N-ratio 为 IPT-PS 所需测试用例数量与 RT 所需测试用例数量的比值，即 $N\text{-ratio}=N_{IPT\text{-}PS}/N_{RT}$. 显然，当 N-ratio$<$1 时，说明 IPT-PS 表现优于 RT；当 N-ratio$>$1 时，说明 IPT-PS 表现劣于 RT.

在 bessj0 的 140 个有效变异中，存在 33 个(23.5%)变异，其对应的 N-ratio$<$1；存在 107 个变异，其对应的 N-ratio$>$1. 在 select 的 8 个有效变异中，存在 6 个(75%)变异，其对应的 N-ratio$<$1；仅有 2 个变异对应的 N-ratio$>$1.

为了进一步分析当 N-ratio$>$1(即 IPT-PS 劣于 RT)的情况下 IPT-PS 与 RT 的性能差别，我们引入了在金融领域广泛应用的条件风险价值(Conditional Value-at-Risk，CVaR)[35]. CVaR 是一种风险计量方法，衡量了当损失超过 VaR 时的平均损失，具体定义如下：

$$CVaR(Pr(r<VaR))=E(r|r\geq VaR).$$

显然，给定风险临界值 VaR，当 $CVaR$ 值越小时，平均损失越小，整体风险越小. 在本文实验中，VaR 与 N-ratio 相关，当 N-ratio$>$1 时，可认为发生损失，因此 $VaR=1$. 以 bessj0 为例，存在 23.5%的变异，其对应的 N-ratio$<$1，于是有 $\Pr(N\text{-ratio}<1)=0.235$，$CVaR(0.235)=E(N\text{-ratio}|N\text{-ratio}\geq 1)=1.7192$. 也就是说，存在 $1\text{-}\Pr(N\text{-ratio}<1)$，即 0.765 的概率使得 N-ratio$\geq$1，所有符合 N-ratio$\geq$1 这一条件的 N-ratio 的均值为 1.7192. 如图 10 所示：x 轴为 N-ratio，y 轴为落入 N-ratio 对应取值范围内的变异的个数. 在 N-ratio$>$1 的 107 个变异中有 70 个变异对应的 N-ratio$\in[1, 1.5]$，$CVaR(0.235)=1.7192$，这表明尽管在 107 个变异中 IPT-PS 的表现较差，但是 IPT-PS 与 RT 的表现差距不大.

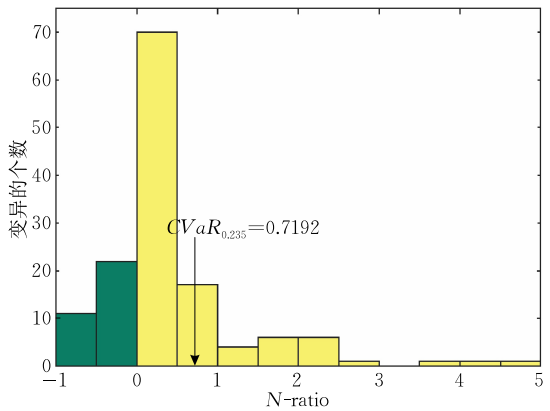


图 10　bessj0 的条件风险价值

通过分析所有 N-ratio$>$1 的变异代码，我们发现：这些变异导致的失效模式是一些组合的复杂失效模式，而并非特定的块状、圆形或点状失效模式，因此，IPT-PS 未能获得较优的失效检测效果. 尽管如此，通过上述两个真实程序及其变异版本的验证，可知 IPT-PS 在真实应用中是可行的，且存在着快速检测到失效的可能.

此外，失效检测能力与待测程序的特性，特别是失效模式有着紧密的联系，然而在真实程序中，失效

模式往往是未知且复杂的,任何一种测试用例生成算法都无法保证在所有失效模式下均表现最优,因此,需要进一步探讨 IPT-PS 的优势应用场景.

于是,我们可以回答问题 3:实验结果表明 IPT-PS 在高维、多种失效模式、真实程序场景下仍然适用,并存在着进一步优化的可能.

综合上述 3 个研究问题可知:IPT-PS 在仅使用接近 RT 时间开销的情况下,可以达到与 ART 相近甚至更好的失效检测能力,权衡了有效性和效率之间的矛盾,是一种高效的测试方法. 同时,IPT-PS 在高维输入域、不同的失效模式以及真实程序中也表现良好,具有广泛的适用性. 此外,实验观察证实了我们的理论分析结论,进一步验证了 IPT、IPT-PS 中关于执行测试用例数与可推断的 θ 取值范围的关系.

5 结 论

随机测试一直被认为是一种最为简便高效的测试用例生成方法,但其算法本身难以实现较好的取样均匀性. 适应性随机测试、划分测试等方法的提出,从测试用例分布的角度上实现了更好的均匀性,失效检测能力在不同场景下实现了提升. 然而,此类方法均需要在时间开销上有着不同程度的牺牲,例如,ART 需要远高于 RT 的生成计算开销;传统划分测试则需要对输入域进行预处理,以获得合理的子域划分,而划分的合理性又往往是决定失效检测能力的关键因素. 因此,一直缺少一种可以在失效检测能力与计算开销之间实现良好平衡的测试用例生成方法.

本文提出了一种基于优先级的迭代划分测试方法 IPT-PS,很好地解决了上述关键性问题. 该方法通过迭代划分、中心采样和基于优先级的用例执行策略实现了测试用例的生成和执行. IPT-PS 结合了上述传统方法的优势,分别解决了其所具有的局限性. 较之上述传统方法,IPT-PS 不需要对输入域进行预处理划分,计算开销极低,更重要的是 IPT-PS 可以实现测试用例的高度均匀分布,使其具有良好的失效检测能力.

本文通过理论分析指出了执行测试用例数与可推断的 θ 取值范围的关系,给出了 IPT-PS 检测失效所需测试用例数量的上界,并通过仿真实验表明,IPT-PS 在仅使用接近 RT 时间开销的情况下,可以达到与 ART 相近甚至更好的失效检测能力.

未来的研究工作主要包括:(1)已有 IPT-PS 方法的改进:探讨是否有其他测试用例的选择方式,使得测试用例在两轮迭代间的分布更加均匀. 同时,考虑如何提高 IPT-PS 在高维空间上性能的稳定性;(2)失效模式的扩展:在已有块状、圆形、点状失效模式的基础上,考虑组合的复杂失效区域、失效区域在输入域中的特定位置(如边界、角落、中心等)对算法性能的影响;(3)实验规模的扩展:本文已采用仿真实验和部分真实程序来验证方法的有效性,后续将增加更多较大规模真实程序的测试,以进一步验证 IPT-PS 方法的有效性.

参 考 文 献

[1] Hamlet R, Random testing//Marciniak J ed. Encyclopedia of Software Engineering. New York, USA: John Wiley & Sons Inc., 1994: 970-978

[2] Myers G J. Art of Software Testing. New York, USA: John Wiley & Sons Inc., 1979

[3] Gutjahr W J. Partition testing vs. random testing: The influence of uncertainty. IEEE Transactions on Software Engineering, 1999, 25(5): 661-674

[4] Chen T Y, Leung H, Mak I K. Adaptive random testing// Proceedings of the 9th Asian Computing Science Conference, LNCS 3321. Chiang Mai, Thailand, 2004: 320-329

[5] Chen T Y, Yu Y T. On the relationship between partition and random testing. IEEE Transactions on Software Engineering, 1994, 20(12): 977-980

[6] Chen T Y, Yu Y T. On the expected number of failures detected by subdomain testing and random testing. IEEE Transactions on Software Engineering, 1996, 22(2): 109-119

[7] Chan F T, Chen T Y, Mak I K, et al. Proportional sampling strategy: Guidelines for software testing practitioners. Information and Software Technology, 1996, 38(12): 775-782

[8] White L J, Cohen E I. A domain strategy for computer program testing. IEEE Transactions on Software Engineering, 1980, 6(3): 247-257

[9] Chan K P, Chen T Y, Towey D. Restricted random testing: Adaptive random testing by exclusion. International Journal of Software Engineering and Knowledge Engineering, 2006, 16(4): 553-584

[10] Chen T Y, Kuo F C, Liu H. Enhancing adaptive random testing through partitioning by edge and center//Proceedings of the 18th IEEE Australian Software Engineering Conference. Melbourne, Australia, 2007: 265-273

[11] Liu Huai, Xie Xiao-Dong, Yang Jing, et al. Adaptive random testing by exclusion through test profile//Proceedings of the 10th International Conference on Quality Software. Zhangjiajie, China, 2010: 92-101

[12] Chen T Y, Kuo F C, Liu H. Adaptive random testing based on distribution metrics. Journal of Systems and Software, 2009, 82(9): 1419-1433

[13] Tappenden A F, Miller J. A novel evolutionary approach for adaptive random testing. IEEE Transactions on Reliability, 2009, 58(4): 619-633

[14] Shahbazi A, Tappenden A F, Miller J. Centroidal voronoi tessellations-a new approach to random testing. IEEE Transactions on Software Engineering, 2013, 39(2): 163-183

[15] Chen T Y, Eddy G R, Merkel G, Wong P K. Adaptive random testing through dynamic partitioning//Proceedings of the 4th International Conference on Quality Software. Braunschweig, Germany, 2004: 79-86

[16] Chen T Y, Huang D H, Zhou Z Q. On adaptive random testing through iterative partitioning. Journal of Information Science and Engineering, 2011, 27(4): 1449-1472

[17] Mayer J. Lattice-based adaptive random testing//Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. Long Beach, USA, 2005: 333-336

[18] Chan K P, Chen T Y, Towey D. Adaptive random testing with filtering: An overhead reduction technique//Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering. Taipei, China, 2005: 292-299

[19] Chen T Y, Kuo F C, Merkel R G, Ng S P. Mirror adaptive random testing. Information and Software Technology, 2004, 46(15): 1001-1010

[20] Ciupa I, Leitner A, Oriol M, Meyer B. ARTOO: Adaptive random testing for object-oriented software//Proceedings of the 30th International Conference on Software Engineering. Leipzig, Germany, 2008: 71-80

[21] Chan K P, Chen T Y, Towey D. Forgetting test cases//Proceedings of the 30th Annual International Computer Software and Application Conference. Chicago, USA, 2006: 485-492

[22] Chen T Y, Kuo F C, Merkel R G, Tse T H. Adaptive random testing: the ART of test case diversity. Journal of Systems and Software, 2010, 83(1): 60-66

[23] Anand S, Burke E, Chen T Y, et al. An orchestrated survey on automated software test case generation. Journal of Systems and Software, 2013, 86(8): 1978-2001

[24] Mayer J, Schneckenburger C. An empirical analysis and comparison of random testing techniques//Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering. Rio de Janeiro, Brazil, 2006: 105-114

[25] Chen T Y, Kuo F C, Towey D, Zhou Z Q. A revisit of three studies related to random testing. Science China Information Sciences, 2015, 58(5): 1-9

[26] Weyuker E J, Jeng B. Analyzing partition testing strategies. IEEE Transactions on Software Engineering, 1991, 17(7): 703-711

[27] Chen T Y, Tse T H, Yu Y T. Proportional sampling strategy: A compendium and some insights. The Journal of Systems and Software, 2001, 58(1): 65-81

[28] Cai K Y, Jing T, Bai C G. Partition testing with dynamic partitioning//Proceedings of the 29th Annual International Computers, Software & Applications Conference, Edinburgh, UK, 2005: 113-116

[29] Cai K Y, Gu B, Hu H, Li Y C. Adaptive software testing with fixed-memory feedback. Journal of Systems and Software, 2007, 80(8): 1328-1348

[30] Lv J P, Hu H, Cai K Y, Chen T Y. Adaptive and random partition software testing. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2014, 44(12): 1649-1664

[31] Chen T Y, Merkel R. An upper bound on software testing effectiveness. ACM Transactions on Software Engineering and Methodology, 2008, 17(3): 1-27

[32] Chen T Y, Kuo F C, Zhou Z Q. On favorable conditions for adaptive random testing. International Journal of Software Engineering and Knowledge Engineering, 2007, 17(6): 805-825

[33] Press W H, Flannery B P, Teukolsky S A, Vetterling W T. Numerical Recipes in C: The Art of Scientific Computing. Cambridge, UK: Cambridge University Press, 1992

[34] Ellims M, Ince D, Petre M. The Csaw C mutation tool: Initial results//Proceedings of the Testing: Academic and Industrial Conference Practice and Research Techniques—MUTATION. Windsor, UK, 2007: 185-192

[35] Rockafellar R T, Uryasev S. Conditional value-at-risk for general loss distributions. Journal of Banking & Finance, 2002, 26(7): 1443-1471

附录 1.　定理证明.

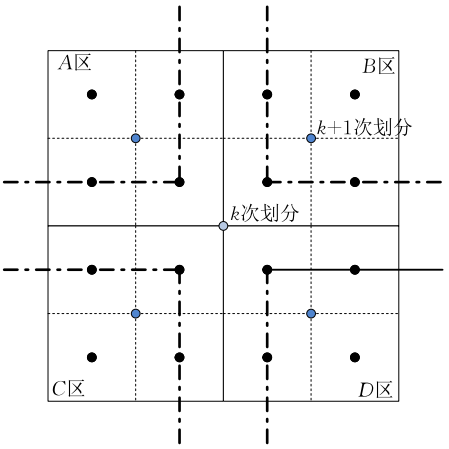
定理 1.　假设输入域和失效域均为 m 维等距连续空间，失效率为 θ. 对于 IPT 有：若进行了 n 次划分，共计执行了 $\sum_{i=0}^{n} 2^{m\cdot i}$ 个测试用例仍无法命中失效域，则可推断 θ 的取值范围为 $\theta<\frac{1}{2^{mn}}$.

证明.　令已执行的且尚未命中失效域测试用例数为 t，切分后新增待执行的测试用例数为 Δt.

(1) 当 $m=2, n=2$ 时，输入域和失效域均为二维平面. 令当前可放入的最大正方形面积为 S. 当前 $t=5$，$\Delta t=2^{2\times 2}=16$，若 t 更新为 $21(1+4+16)$，此时 $S<1/16$，可推断 θ 的取值范围为 $\theta<\frac{1}{2^{mn}}=\frac{1}{16}$ 成立.

(2) 假定当 $m=2, n=k$ 时，$t=1+2^2+\cdots+2^{2(k-1)}$，划分后的子域为 2^{2k} 个，即 $\Delta t=2^{2k}$，若 t 更新为 $1+2^2+\cdots+2^{2(k-1)}+2^{2k}$，此时 $S<\frac{1}{2^{2k}}$，即 $\theta<\frac{1}{2^{2k}}$ 成立. 当 $n=k+1$ 时，如附图 1 所示，在完成 k 次划分后产生了 $A\sim D$ 四个区域，各区域面积均为 $\frac{1}{2^{2k}}$，在 $k+1$ 次划分后，输入域将划分为 $2^2\times 2^{2k}$ 个子域，这些子域的中点成为新增待执行的测试用例，对于

$A\sim D$ 区域内而言，每个区域新增 2^2 个待执行测试用例，则可以确保每个区域内可放入的最大正方形面积不超过 $\frac{1}{2^2}\times\frac{1}{2^{2k}}$. 接下来我们考虑 $A\sim D$ 四个区域的相连区域(即图中虚线框住的十字型区域)中是否需要放入新的测试用例. 显然，无需新增测试用例，相连区域中能够放入的最大正方形面积已经小于 $\frac{1}{2^2}\times\frac{1}{2^{2k}}$. 因此，$t$ 更新为 $1+2^2+\cdots+2^{2(k-1)}+2^{2k}+2^{2(k+1)}$ 时，可推断 θ 的取值范围为 $\theta<\frac{1}{2^{2(k+1)}}$ 成立.



附图 1 定理 1 证明示意图

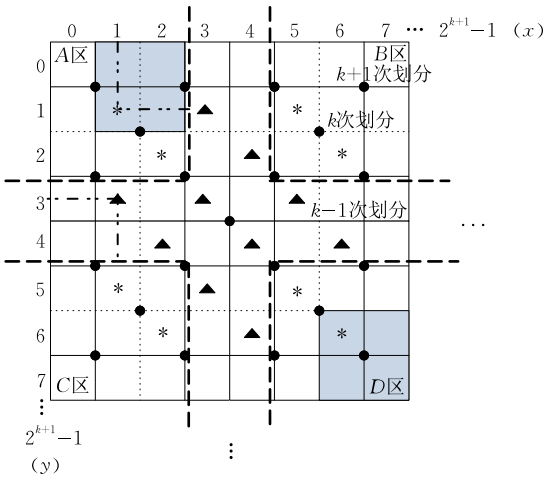
(3) 假定当 $m=k'$，对于任意 n，上述定理均成立. $m=k'+1$ 时，输入域和失效域均为 $k'+1$ 维超立方体. 当进行 n 次划分时，可以分解为首先对 k' 维超立方体进行 n 次划分产生 $2^{k'n}$ 个子域，再对最后 1 维进行 n 次划分产生 2^n 个子域. 因此 $k'+1$ 维划分后的子域数量为 $2^{k'n}\times 2^n$，需新增 $2^{k'n}\times 2^n$ 个待执行测试用例，以确保能够放入的最大 $k'+1$ 维超立方体体积将小于当前划分的子域体积 $\frac{1}{2^{k'n}\times 2^n}$. 因此，$t$ 更新为 $1+2^{(k'+1)\cdot 1}+\cdots+2^{(k'+1)\cdot n}$ 时，可推断 θ 的取值范围为 $\theta<\frac{1}{2^{(k'+1)n}}$ 成立. 证毕.

定理 2. 假设输入域和失效域均为 m 维等距连续空间，失效率为 θ. 对于 IPT-PS 有：若进行了 n 次划分，共计执行了 $\sum_{i=0}^{n-1}2^{m\cdot i}+\frac{(2^n-2)^m}{2^{m-1}}$ 个测试用例仍无法命中失效域，则可推断 θ 的取值范围为 $\theta<\frac{1}{2^{m(n-1)}}\left(\frac{3}{4}\right)^m$.

证明. 令已执行的且尚未命中失效域测试用例数为 t，切分后新增待执行的测试用例数为 Δt，其中优先级最高的集合 T_1 中的测试用例数为 Δt_1.

(1) 当 $m=2,n=2$ 时，输入域和失效域均为二维平面. 当前 $t=5,\Delta t=16$，其中 $\Delta t_1=\frac{(2^n-2)^m}{2^{m-1}}=\frac{(2^2-2)^2}{2^{2-1}}=2$ 成立. 当 t 更新为 $t+\Delta t_1$，即 $t=7$ 时，可推断 θ 的取值范围为 $\theta<\frac{1}{2^{m(n-1)}}\left(\frac{3}{4}\right)^m=\frac{1}{2^{2(2-1)}}\left(\frac{3}{4}\right)^2=\frac{9}{64}$ 成立.

(2) 假定当 $m=2,n=k$ 时，$t=1+2^2+\cdots+2^{2(k-1)}$，$\Delta t=2^{2k}$，其中 $\Delta t_1=\frac{(2^k-2)^2}{2^{2-1}}$ 成立，当 t 更新为 $1+2^2+\cdots+2^{2(k-1)}+\Delta t_1$ 时，可推断 θ 的取值范围为 $\theta<\frac{1}{2^{2(k-1)}}\left(\frac{3}{4}\right)^2$ 成立. 当 $n=k+1$ 时，如附图 2 所示，在完成 k 次划分后，各区域面积均为 $\frac{1}{2^{2k}}$(如图中阴影面积所示)，在 $k+1$ 次划分后，输入域将划分为 $2^2\times 2^{2k}$ 个子域，这些子域的中点成为新增待执行的测试用例，$\Delta t=2^{2(k+1)}$. 对于 $k-1$ 次划分后产生的 $2^{2(k-1)}$ 个子域，以区域 A 为例，每个区域新增“$*$”测试用例，则可以确保每个区域内可放入的最大正方形面积变更为 $\frac{3}{4}\times\frac{3}{4}\times\frac{1}{2^{2k}}$.



附图 2 定理 2 证明示意图

接下来我们考虑与区域 A 相邻的共计 4 个区域形成的相连区域(即图中虚线框住的十字型区域)是否需要放入新的测试用例. 显然，必须新增“▲”测试用例才能确保相连区域中能够放入的最大正方形面积也小于 $\frac{3}{4}\times\frac{3}{4}\times\frac{1}{2^{2k}}$. 于是 $\Delta t_1=\frac{(2^{k+1}-2)^2}{2^{2-1}}$ 成立，当 t 更新为 $1+2^2+\cdots+2^{2k}+\Delta t_1$ 时，可推断 θ 的取值范围为 $\theta<\frac{1}{2^{2k}}\left(\frac{3}{4}\right)^2$ 成立.

(3) 假定当 $m=k'$，对于任意 n，上述定理均成立. 当 $m=k'+1$ 时，进行 n 次划分时，可以分解为首先对 k' 维超立方体进行 n 次划分，有 $\Delta t_1=\frac{(2^n-2)^{k'}}{2^{k'-1}}$，再对最后 1 维进行 n 次划分，有 $\Delta t_1'=\frac{2^n-2}{2}$，共需新增 $\Delta t_1\,\Delta t_1'=\frac{(2^n-2)^{k'+1}}{2^{k'}}$ 个待执行测试用例. 此时可放入的最大 $k'+1$ 维超立方体体积为 k' 维超立方体体积 $\frac{1}{2^{k'\cdot(n-1)}}\left(\frac{3}{4}\right)^{k'}$ 乘以最后 1 维进行 n 次划分后可放入的最大线段长度 $\frac{1}{2^{n-1}}\times\frac{3}{4}$，即 $\frac{1}{2^{(k'+1)\cdot(n-1)}}\left(\frac{3}{4}\right)^{k'+1}$. 因此，当 $t=\sum_{i=0}^{n-1}2^{(k'+1)\cdot i}+\frac{(2^n-2)^{k'+1}}{2^{k'}}$ 时，可推断 θ 的取值范围为 $\theta<\frac{1}{2^{(k'+1)(n-1)}}\left(\frac{3}{4}\right)^{k'+1}$ 成立. 证毕.

ZHANG Xiao-Fang, born in 1980, Ph. D., associate professor. Her research interests include software testing and analysis, fault localization, and reinforcement learning.

ZHANG Zong-Zhang, born in 1985, Ph. D., associate professor. His research interests include POMDPs, reinforcement learning and multi-agent systems.

XIE Xiao-Yuan, born in 1983, Ph. D., professor. Her research interests include software testing and analysis, fault localization, debugging and search-based software engineering.

ZHOU Yi-Cheng, born in 1990, M. S. candidate. His research interests include software engineering and reinforcement learning.

Background

Test case generation has been a popular research area in software testing. This paper addresses the problem from black-box perspective, where diversity and evenly spreading of the test cases have always been two challenging issues. Commonly adopted solutions include random testing, partition-based testing, adaptive random testing, etc. However, they all have weakness. For example, random testing is high in efficiency but low in fault detection ability. Adaptive random testing, on the other hand, has better effectiveness, but normally requires much higher computational cost. While for current partition-based testing methods, there is usually much preprocess effort required. Our method outperforms these solutions in terms of having high efficiency and fairly good fault detection ability, as well as a dynamic partition strategy that does not require any work prior to the test case generation. Specifically, we propose an Iterative Partition Testing based on Priority Sampling algorithm, IPT-PS, which iteratively divides the input domain into grids, and select the center point of each grid as test case. Priority-based execution strategy is then applied on newly generated test cases in each round of iteration. We theoretically prove the upper bound of the effectiveness with our method, and conduct comprehensive empirical analysis. Our experimental results show that IPT-PS can achieve high effectiveness with quite low time cost.

This work is partially supported by the National Natural Science Foundation of China (61103045, 61502329, 61502323, 61572375) and the Collaborative Innovation Center of Novel Software Technology and Industrialization. These projects aim to solve the testing and debugging problem in large-scaled software system, enrich the applications of machine learning techniques in software testing. The authors have been working in the related areas for many years and published papers in various top international conferences and journals, including TOSEM, JSS, IST, ICSE, COMPSAC, SEKE, AAAI, ICML, UAI, ICAPS, etc.