第四届中国基于搜索的软件工程研讨会

南京

2015.6.13

进化变异测试

巩敦卫

中国矿业大学

dwgong@vip.163.com

主要内容

1. 变异测试
2. 进化测试
3. 我们的工作
4. 研究展望

主要内容

变异测试

进化测试

我们的工作

研究展望

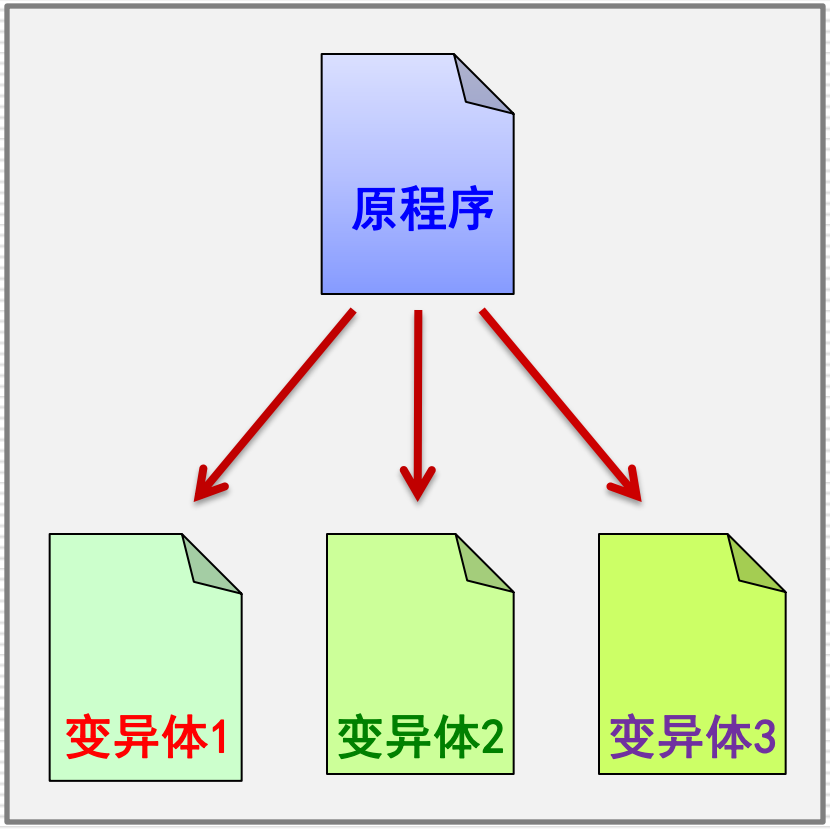
测试用例集性能

测试用例集

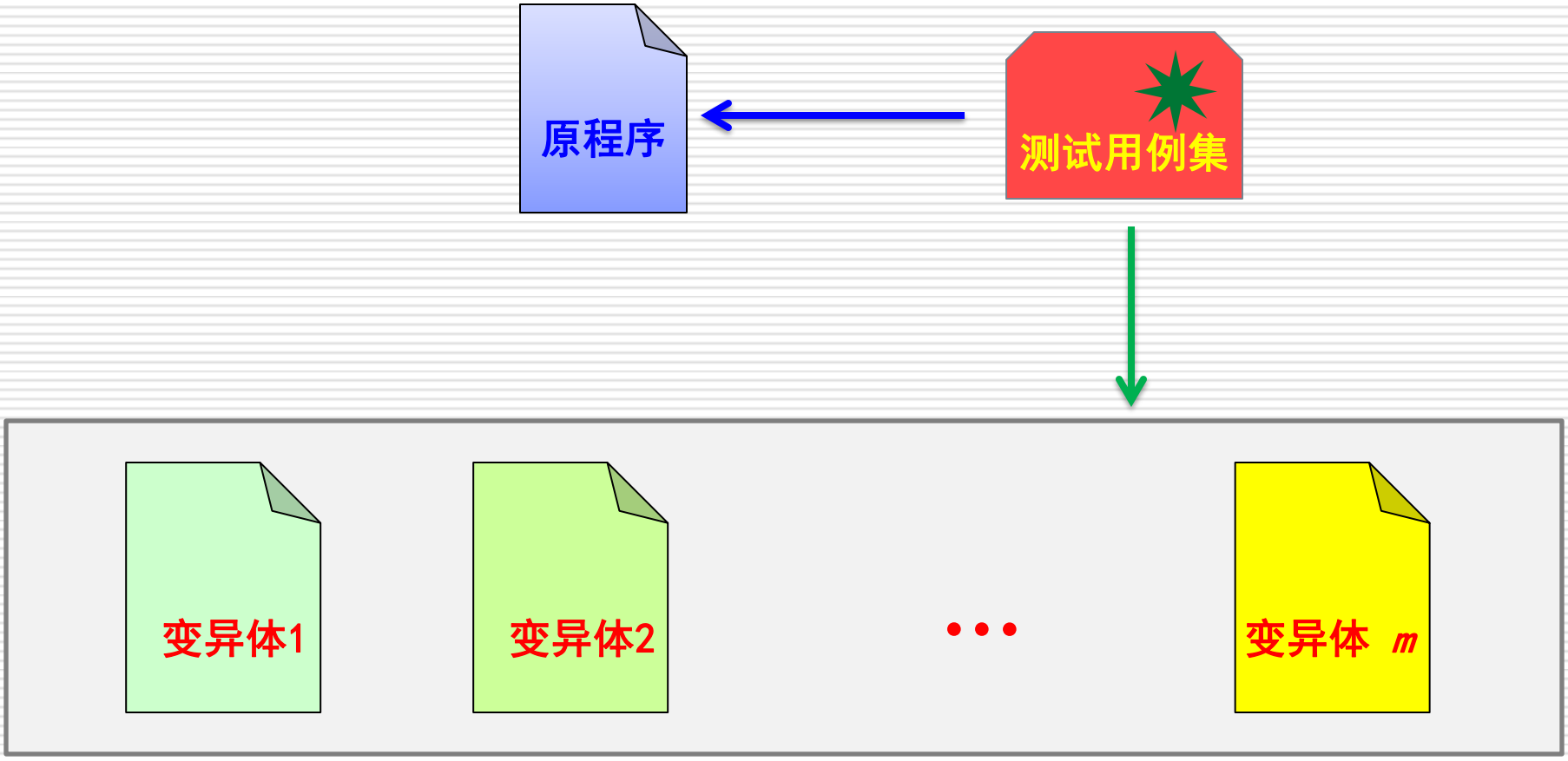
测试用例集检测缺陷的能力如何？

```
max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

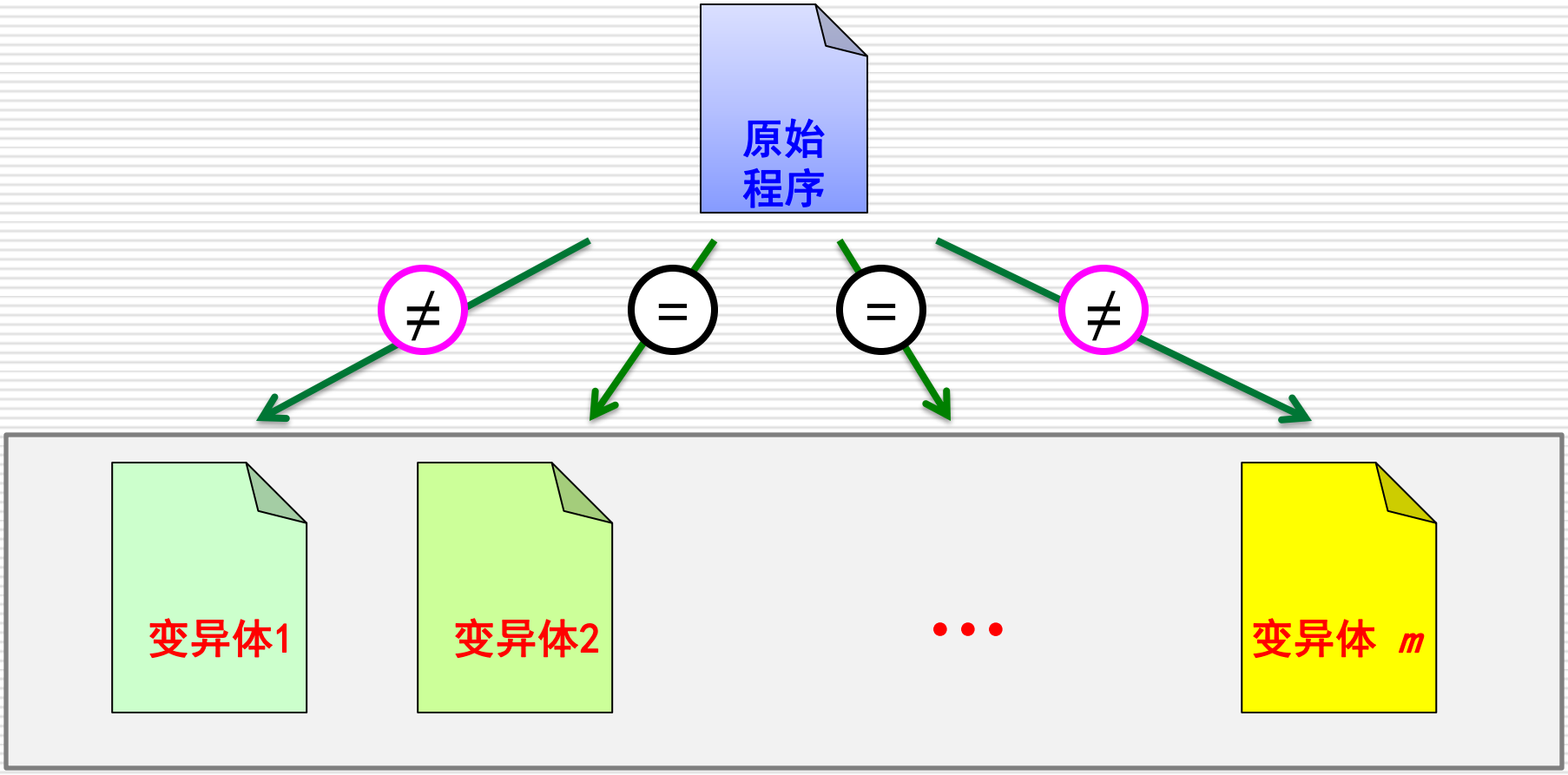
a >= b
a < b
a <= b



程序执行



程序输出比较

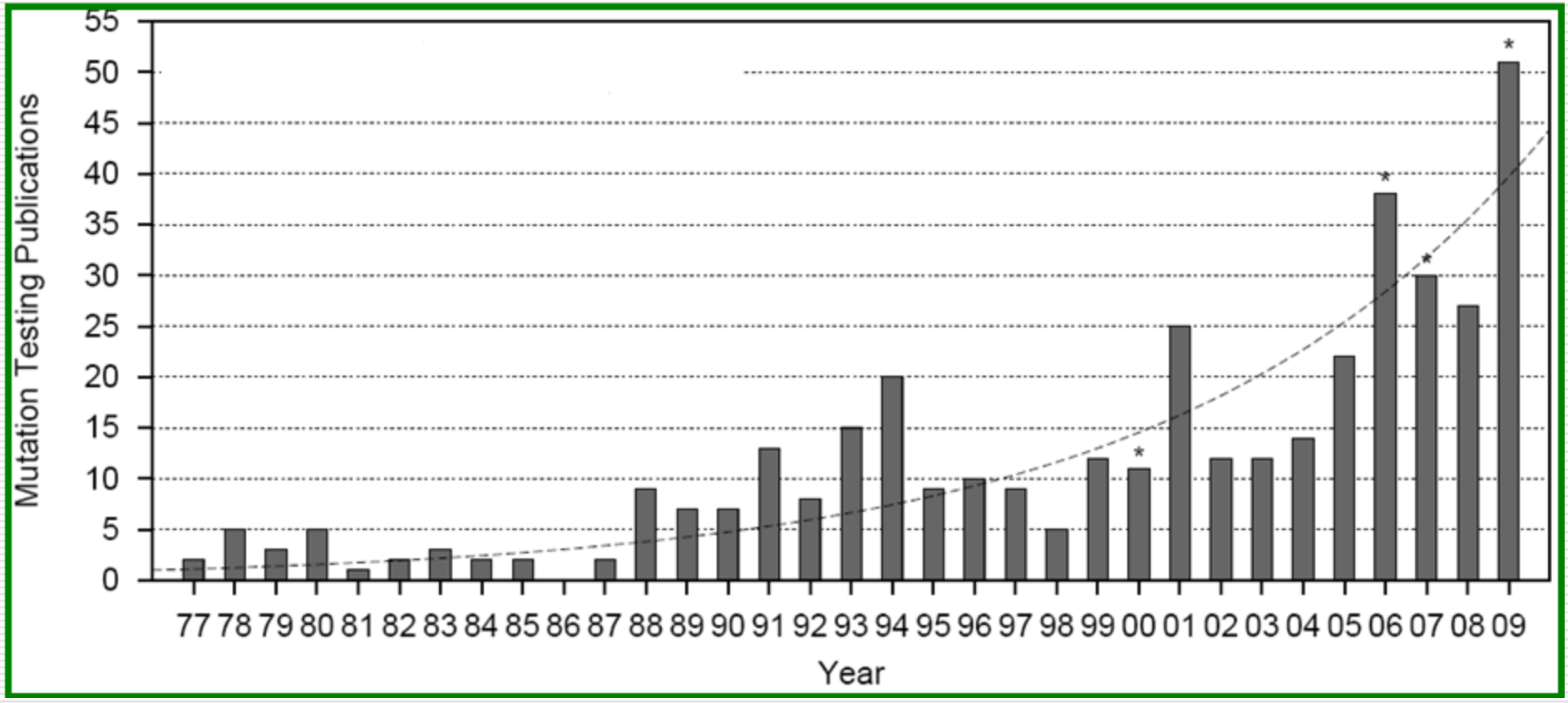


变异测试用例评价

- 如果存在某个测试用例，能够从输出结果上区分原程序和变异体，称该变异体是被杀死的
- 暂时没被杀死的变异体，称为活着的变异体
- 任何测试用例都不能杀死的变异体，称为等价变异体

变异得分=被杀死的变异体个数/非等价变异体个数

变异测试论文（1977-2009）



Jia Y, Harman M. An analysis and survey of the development of mutation testing[J]. IEEE Transactions on Software Engineering, 2011, 37(5): 649-678.

应用与问题

- 变异测试应用
  - 评价测试用例集质量
  - 辅助生成测试用例集
  - 修正软件缺陷

- 存在问题
  - 测试代价大
  - 等价变异体难以判定

主要内容

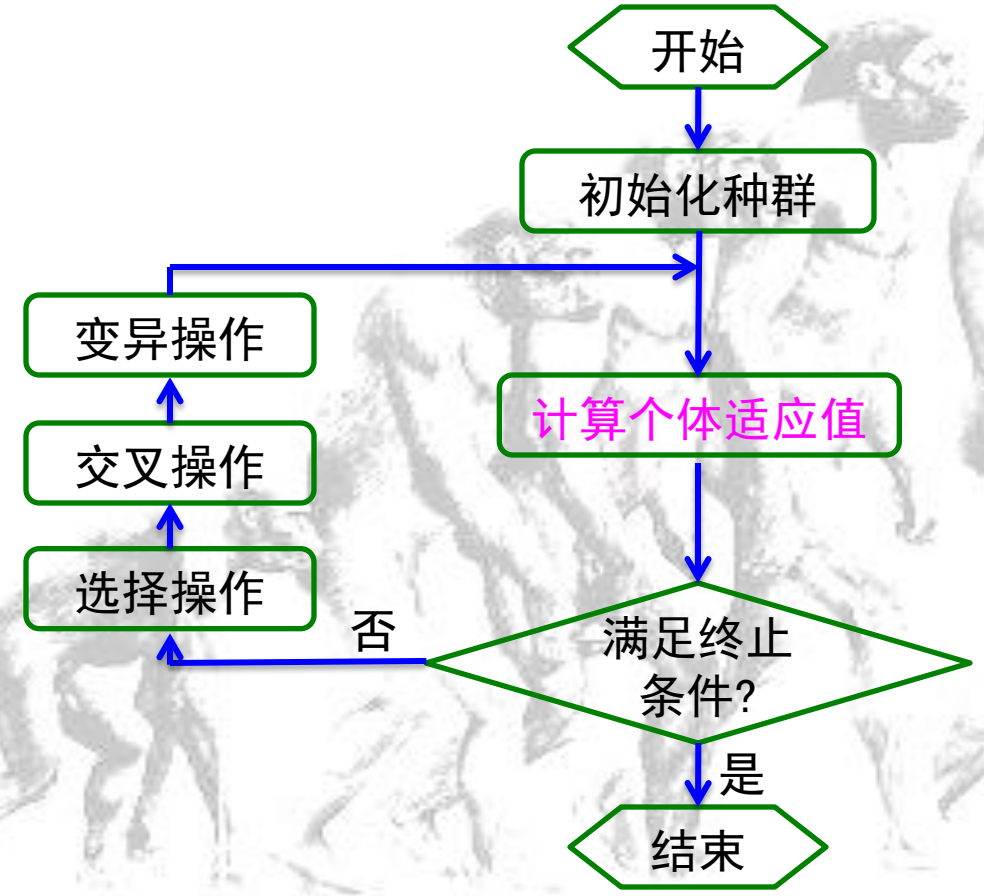
1 变异测试

2 进化测试

3 我们的工作

4 研究展望

遗传算法



遗传算法生成测试用例

- 测试用例生成问题转化为函数优化问题
- 设计适应度函数
- 对测试用例编码
- 解码并执行被测程序，评价测试用例性能
- 进化一定代数后，生成期望测试用例

- 转化方法

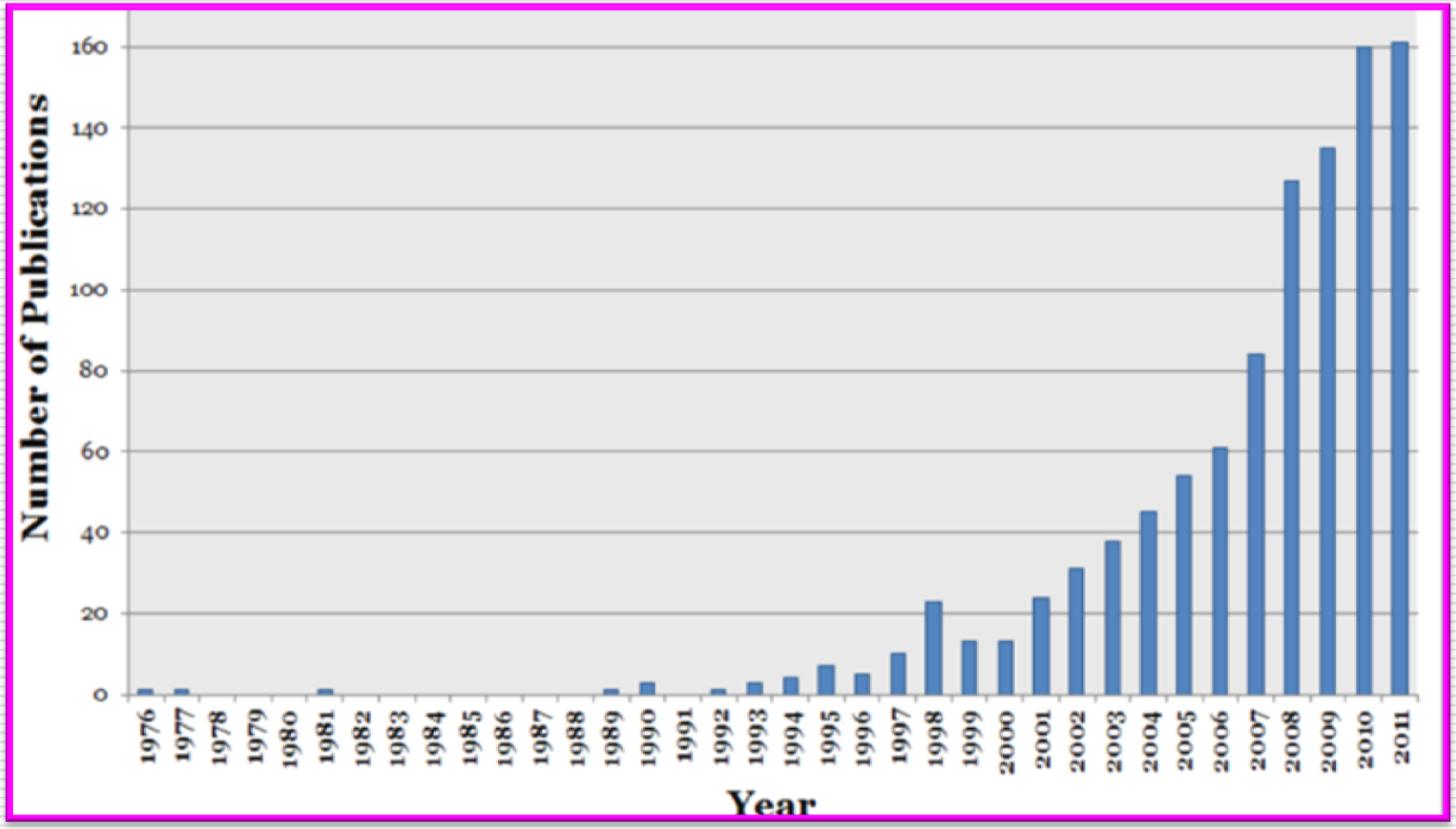
```
if(x >= 21)
{
   目标语句
}
```

$$\min f(x)=\begin{cases}21-x & x<21\\ 0 & \text{其它}\end{cases}$$

基于其它搜索方法的测试用例生成

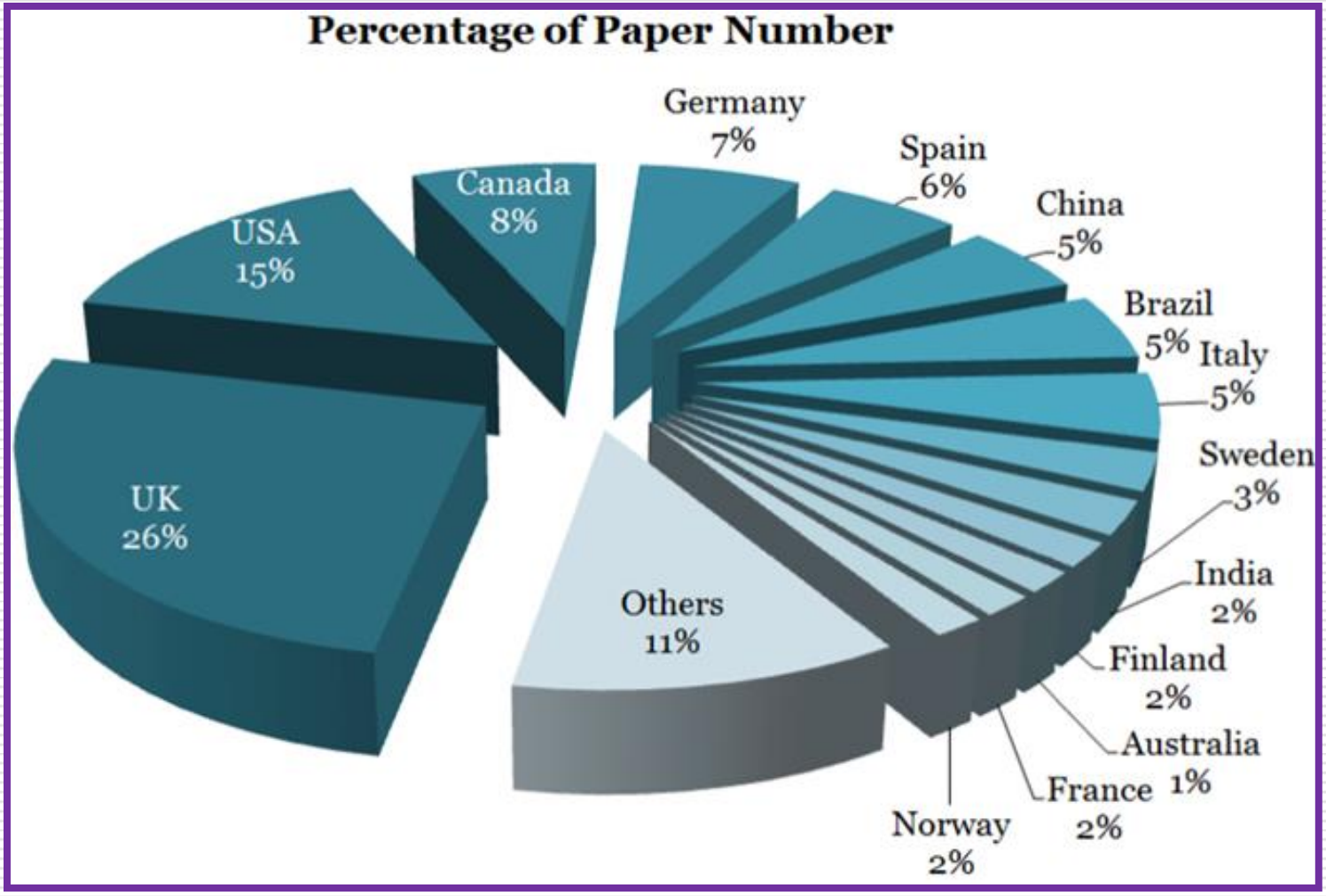
- 改变变量法
- 爬山法
- 模拟退火算法
- 禁忌搜索方法
- 微粒群优化方法
- …

基于搜索软件工程研究进展-论文



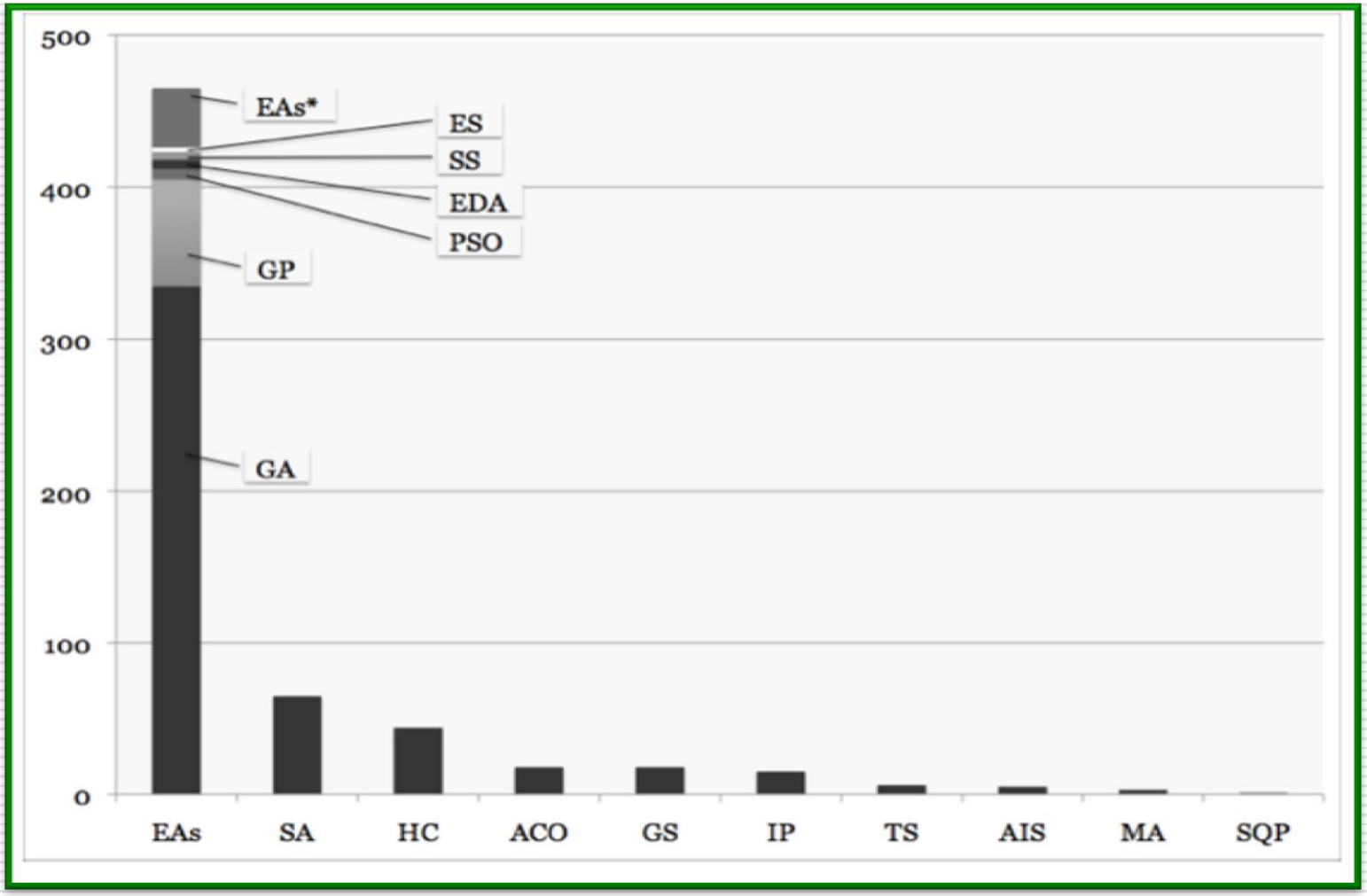
论文数量

基于搜索软件工程研究进展-地区



地区分布

基于搜索软件工程研究进展-算法



算法分布

基于搜索软件工程专刊

- Journal of Software Maintenance and Evolution (JSME)
- IEEE Transactions on Software Engineering(TSE)
- Software: Practice and Experience(SPE)
- Empirical Software Engineering(EMSE)
- Journal of Systems and Software (JSS)
- Information and Software Technology (IST)
- Computers and Operations Research (COR)
- 软件学报

软件学报基于搜索软件工程专刊

基于搜索软件工程研讨会

第四届基于搜索软件工程研讨会

- 地点：南京大学
- 时间：6月13日召开
- 网址：http://gist.nju.edu.cn/csbse15/#

主要内容

1 变异测试

2 进化测试

3 我们的工作

4 研究展望

遗传算法用于变异测试

- 基于占优关系约简变异体
- 采用集合进化生成测试用例

遗传算法用于变异测试

- 基于占优关系约简变异体
- 采用集合进化生成测试用例

弱变异测试转化

- 弱变异测试准则：执行变异语句之后状态发生变化
- 原程序P的一条语句s，s'为s的一条变异语句。以$s!=s'$为条件，构造分支语句b

覆盖b真分支的测试用例，能够杀死b对应的变异体m

变异测试问题 → 分支覆盖问题

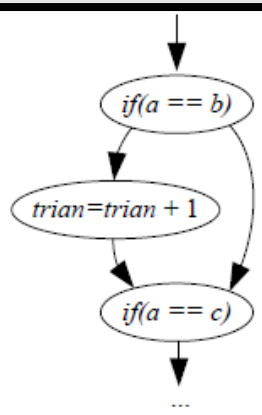
新程序形成

- $M=\{m_1,m_2,\ldots,m_{|M|}\}$，$B=\{b_1,b_2,\ldots,b_{|B|}\}$，$|M|=|B|$，$m_i$和$b_i$一一对应，$b_i$称为变异分支
- 所有变异分支集成到原程序P中，形成新的被测程序P'

示例程序

```
1  public static int getTri(int a, int b, int c) {
2     int trian;
3     if (a <= 0 || b <= 0 || c <= 0) {
4        return 4;
5     }
6     trian = 0;
7     if (a == b) {
8        trian = trian + 1;
9     }
10    if (a == c) {
11       trian = trian + 2;
12    }
13    if (b == c) {
14       trian = trian + 3;
15    }
16    if (trian == 0) {
17       if (a + b <= c || b + c <= a || a + c <= b) {
18          return 4;
19       } else {
20          return 1;
21       }
22    }
23    if (trian > 3) {
24       return 3;
25    } else if (trian == 1 && a + b > c) {
26       return 2;
27       } else if (trian == 2 && a + c > b) {
28          return 2;
29       } else if (trian == 3 && b + c > a) {
30          return 2;
31    }
32    return 4;
33 }
```

(a)



(b)

original: *if(a == b)*

mutant branches:

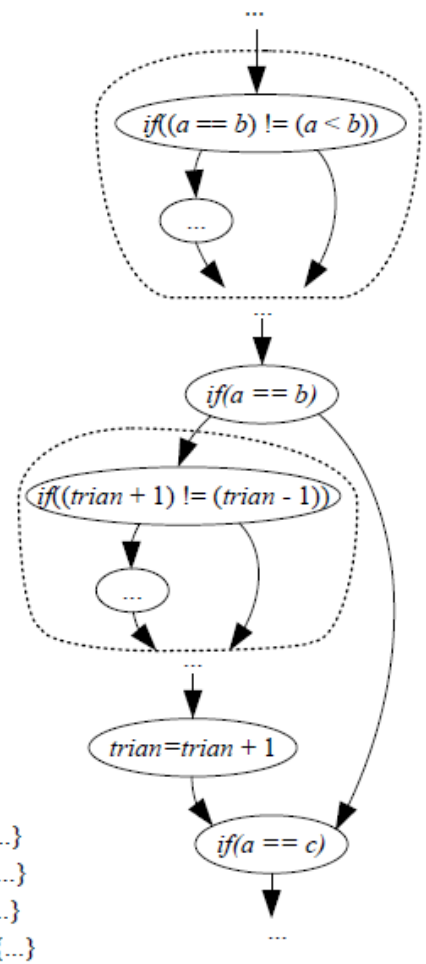
if((a == b) != (a < b)){...}
if((a == b) != (a <= b)){...}
if((a == b) != (a != b)){...}
if((a == b) != (a >= b)){...}
if((a == b) != (a > b)){...}

original: *trian = trian + 1*

mutant branches:

if((trian + 1) != (trian - 1)){...}
if((trian + 1) != (trian * 1)){...}
if((trian + 1) != (trian / 1)){...}
if((trian + 1) != (trian % 1)){...}

(c)



(d)

产生的问题

- 转化后变异分支很多
- 所有变异分支集成到P'中，增加了转化后程序的复杂度

占优关系

- 程序的部分语句之间存在相关性
- 新程序P'的部分变异分支之间同样存在相关性
- 通过相关性分析，有可能减少需要覆盖的变异分支，从而减少需要杀死的变异体

- 对于P'的2个变异分支b_i和b_j，如果对于任意测试用例，b_i的真分支执行，b_j的真分支必然执行，称b_i 占优b_j，记为 $b_i \succ b_j$ 。b_i称为占优分支，b_j称为被占优分支
- 对于 $b_i \in B$，如果不存在任何 $b_j \in B, b_j \neq b_i$ ，使得 $b_j \succ b_i$ 成立，称b_i为非被占优分支。所有非被占优分支构成的集合，称为非被占优分支集，记为B^{nd}

基于占优关系约简变异体

P'的占优关系图是一个有向图，记为$D(P')=\{V(B), E(B)\}$

- ✓ $V(B)$是顶点集，$E(B)$是边集
- ✓ 对于 $\forall b_i, b_j \in B, b_i \neq b_j$ ，且 $b_i \succ b_j$ ，那么，$< v_i, v_j > \in E(B)$

$D(P')$ 中，入度为0的顶点对应的变异分支构成的集合，是非被占优分支集B^{nd}

B^{nd}对应的变异体，是约简后的变异体

需要验证的问题

- 所提方法能否有效约简变异体？
- 变异体约简后，测试用例生成效率能否提高？
- 变异体约简后，生成测试用例的有效性如何？
- 生成测试用例的缺陷检测能力有何变化？

被测基准程序

ID	Name	LOC	Function
J1	Triangle	35	Return the type of a triangle by three integer inputs.
J2	Euclid	12	Euclid's algorithm to find the greatest common divisor of two integers.
J3	Mid	26	Return the mid value of three integers.
J4	Bubble	16	Bubble sort algorithm.
J5	Trash_And_Take_Out	26	Not reported.
J6	Cal	50	Calculate the number of days between the two dates in the same year.
J7	Bank_Account	34	Simulates bank account deposit and withdrawal services.
J8	Smoke Detector	40	Detects the current room smoke level by double inputs.
J9	Vending Machine	112	Vend a small number of products through coins inserting, items choosing and changes getting.
J10	Sort_Code	70	Validates the UK bank sort_code of the form XX-XX-XX where X is an integer digit.

变异体约简率

ID	No. of mutation branches	No. of infeasible branches	No. of branches after reduction	Reduction rate (%)
J1	325	22	67	77.89
J2	57	11	10	78.26
J3	115	14	27	73.27
J4	92	5	16	81.61
J5	112	22	7	92.22
J6	316	43	54	80.22
J7	97	9	14	84.09
J8	175	19	33	78.85
J9	492	35	95	79.21
J10	209	21	37	80.30
		Average		80.59

测试用例生成效率

ID	Before Reduction			After Reduction			Speedup
	No. of tests	Time(ms)	Time per test(ms)	No. of tests	Time(ms)	Time per test(ms)	
J1	33.8	4.334284	0.128233	23.6	0.650308	0.027555	6.665
J2	5.7	0.439317	0.077073	3.8	0.220242	0.057958	1.995
J3	16.5	3.840068	0.232731	9.8	0.699357	0.071363	5.491
J4	5.8	10.002405	1.724553	4.2	3.656434	0.87058	2.736
J5	6.8	1.394492	0.205072	4.7	0.297708	0.063342	4.684
J6	25.7	169.806308	6.607249	17.5	5.194644	0.296837	32.689
J7	13.4	1.010467	0.075408	9.5	0.329932	0.03473	3.063
J8	11.8	2.028524	0.171909	9.8	0.661253	0.067475	3.068
J9	44.5	4698.295507	105.579674	28.4	189.996382	6.690013	24.728
J10	16.9	6.657047	0.393908	12.5	1.686543	0.134923	3.947
Average							8.907

测试用例有效性

ID	No. of mutants	No. of equivalent mutants	Before reduction		After reduction	
			No. of Killed mutants	Mutation score (%)	No. of Killed mutants	Mutation score (%)
J1	325	40	282	98.95	280	98.25
J2	57	12	45	100.00	45	100.00
J3	115	18	97	100.00	97	100.00
J4	92	7	85	100.00	85	100.00
J5	112	29	82	98.80	82	98.80
J6	316	43	273	100.00	266	97.44
J7	97	13	83	98.81	83	98.81
J8	175	20	155	100.00	155	100.00
J9	492	36	449	98.46	446	97.81
J10	209	40	168	99.41	168	99.41

测试用例缺陷检测能力

ID	After reduction					Before reduction				
	30%	60%	90%	Top	No. of detected faults per test	30%	60%	90%	Top	No. of detected faults per test
J1	2	4	12	24	11.9	6	8	24	34	8.4
J2	1	2	3	4	11.3	1	2	5	6	7.5
J3	2	4	7	9	10.8	2	7	14	15	6.5
J4	1	1	3	5	17	1	3	5	7	12.1
J5	1	2	3	5	16.6	1	4	6	8	10.4
J6	1	5	13	16	17.1	1	11	22	24	11.4
J7	1	2	6	8	10.5	4	5	10	12	7
J8	1	3	5	9	17.2	1	6	10	13	11.9
J9	3	10	16	25	18.2	4	12	28	40	11.4
J10	1	3	5	11	15.4	5	8	13	17	9.9
Average					14.6	Average				9.7

开源工业程序

ID	Class	LOC	Methods		Mutants		From package
			Total	Testable	Total	Testable	
I1	Word_Utils	173	12	2	262	243	org.apache.commons.lang3.text
I2	Duration_Format_Utils	365	9	3	722	576	org.apache.commons.lang3.time
I3	Help_Format	416	39	4	586	477	org.apache.commons.cli
I4	Number_Utils	636	47	33	1622	1406	org.apache.commons.lang3.math
I5	Unix_Crypt	311	12	9	1123	1097	org.apache.commons.codec.digest
I6	Md5Crypt	107	7	2	169	158	org.apache.commons.codec.digest
	Sum	2008	126	53	4484	3957	

约简率

ID	No. of mutant branches	No. of infeasible mutant branches	No. of equivalent mutants	No. of mutant branches after reduction	Reduction Rate (%)
I1	243	26	34	36	79.72
I2	576	52	61	83	82.44
I3	477	35	44	91	77.38
I4	1406	184	237	293	71.69
I5	1097	167	209	131	81.40
I6	158	12	12	30	79.45
Sum	3957	476	597	664	Average = 78.68

约简前后生成测试用例的性能

约简前

ID	No. of tests	Covered Mutant Branches	Mutant Branches Coverage (%)	Killed Mutants (%)	Mutation Score (%)
I1	9.7	213	98.16	203	97.13
I2	24.7	523	99.81	501	97.28
I3	23.9	434	98.19	426	98.38
I4	102.5	1208	98.85	1126	96.32
I5	9.4	903	97.1	859	96.73
I6	6.8	146	100	146	100
	Sum = 177	Sum = 3427	Average = 98.69	Sum = 3261	Average = 97.64

约简后

ID	No. of tests	No. of covered mutant branches	Mutant Branches Coverage (%)	No. of killed mutants	Mutation score (%)
I1	4.8	211	97.24	199	95.22
I2	19.3	518	98.85	498	96.70
I3	18.9	429	95.15	420	97.00
I4	79.8	1178	96.4	1117	95.55
I5	5.7	901	96.88	852	95.95
I6	4.2	146	100	146	100
	Sum = 132.7	Sum = 3383	Average = 97.42	Sum = 3232	Average = 96.74

小结

- 提出基于占优关系的变异体约简新方法
  - ✓ 构建变异分支，集成到原程序中，形成新的被测程序
  - ✓ 分析新程序中变异分支之间的占优关系
  - ✓ 占优关系图中入度为0的顶点，对应约简后的变异体
- 实验验证所提方法的有效性
  - ✓ 生成测试用例少
  - ✓ 生成测试用例的效率高
  - ✓ 基于强变异测试准则有效
  - ✓ 单测试用例的缺陷检测能力强

遗传算法用于变异测试

- 基于占优关系约简变异体
- 采用集合进化生成测试用例

采用集合进化生成测试用例

- 目的
  - ✓ 高效生成具有高缺陷检测能力的测试用例集

- 思想
  - ✓ 基于新被测程序的变异分支，建立分支覆盖测试用例生成问题的数学模型
  - ✓ 设计测试用例集进化个体的适应度函数
  - ✓ 设计测试用例集进化个体的遗传策略
  - ✓ 采用集合进化方法求解

测试用例生成问题模型

- 被测程序需要覆盖的变异分支集为B，包含$|B|$个变异分支
- 测试用例 $\vec{x} \in D$，D为输入域
- 对于变异分支$b_i \in B$，$\vec{x}$对b_i的分支距离为 $f_i(\vec{x})$
- 测试用例集 $X = (\vec{x}_1, \vec{x}_2, ..., \vec{x}_m), F_i(X) = \min\limits_{j=1,2,\cdots,m} \{f_i(\vec{x}_j)\}$

$$\min F(X) = (F_1(X), F_2(X), ..., F_{|B|}(X))$$
$$\text{s.t. } X = (\vec{x}_1, \vec{x}_2, ..., \vec{x}_m),\ \vec{x}_k \in D,\ k = 1, 2, ..., m$$

- 决策变量包含若干测试用例
- 以X覆盖B的所有变异分支作为优化目标

集合进化策略

- 适应度函数 $Fit(X)=1-e^{-\sum_{i=1}^{|B|}F_i(X)}$

- 个体之间交叉

$$X^1=(\vec{x}_1^{\,1},\vec{x}_2^{\,1},...,\vec{x}_k^{\,1},...,\vec{x}_m^{\,1}) \quad X^{1'}=(\vec{x}_1^{\,1},\vec{x}_2^{\,1},...,\vec{x}_k^{\,2},...,\vec{x}_m^{\,2})$$

$$X^2=(\vec{x}_1^{\,2},\vec{x}_2^{\,2},...,\vec{x}_k^{\,2},...,\vec{x}_m^{\,2}) \quad X^{2'}=(\vec{x}_1^{\,2},\vec{x}_2^{\,2},...,\vec{x}_k^{\,1},...,\vec{x}_m^{\,1})$$

- 个体内部交叉

$$\vec{x}_i^{\,1}=(x_i^{11},x_i^{12},...,x_i^{1k},...,x_i^{1l}) \quad \vec{x}_i^{\,1'}=(x_i^{11},x_i^{12},...,x_j^{1k},...,x_j^{1l})$$

$$\vec{x}_j^{\,1}=(x_j^{11},x_j^{12},...,x_j^{1k},...,x_j^{1l}) \quad \vec{x}_j^{\,1'}=(x_j^{11},x_j^{12},...,x_i^{1k},...,x_i^{1l})$$

集合进化策略

- 个体变异

  对于个体X^1，选择测试用例 $\vec{x}_i^{\,1}$ ，生成新的用例 $\vec{x}_i^{\,1'} \in D$，

  且$\vec{x}_i^{\,1'} \neq \vec{x}_i^{\,1}$；用 $\vec{x}_i^{\,1'}$代替X^1中的$\vec{x}_i^{\,1}$，产生新个体 $X^{1'}$

- 个体选择

  计算个体适应值，选择适应值最小的若干个体，形成下一代种群

算法步骤

Step1：设置集合进化控制参数

Step2 ： 初始化种群

Step3：判断终止条件是否满足？如果满足，转步骤6

Step4：计算个体适应值

Step5：实施集合进化操作，生成下一代种群，转步骤3

Step6：终止集合进化，输出测试用例及其杀死的变异体

需要验证的问题

- 所提方法能否降低测试用例生成成本？
- 生成的测试用例集能否基于弱变异测试准则有效杀死变异体？
- 生成的测试用例集能否基于强变异测试准则有效杀死变异体？

被测基准程序

ID	程序	行数	方法		变异体个数	说明
			总数	被测试个数		
J1	TranshAndOut	30	2	2	111	未知
J2	Mid	26	1	1	115	3个整数的最小值
J3	FourBalls	28	1	1	213	4个球体的相对重量
J4	Triangle	36	1	1	325	三角形类型判定
J5	Cal	46	2	2	314	两个日期之间天数
J6	MD5Crypt	107	7	2	158	org.apache.commons.codec.digest
J7	DurationFormatUtils	365	9	1	377	org.apache.commons.lang3.time
J8	HelpFormat	416	39	2	301	org.apache.commons.cli
J9	UnixCrypt	311	12	5	805	org.apache.commons.codec.digest
J10	WordUtils	173	12	2	243	org.apache.commons.lang3.text
J11	NumberUtils	636	47	21	912	org.apache.commons.lang3.math
J12	PatternOptionBuilder	96	3	3	204	org.apache.commons.cli
J13	FieldUtils	142	15	3	242	org.joda.time.field
	Sum.	2412	151	46	4320	

测试用例生成成本

ID	GA			本文方法			比较	
	迭代次数	时间(ms)	测试用例个数	迭代次数	时间(ms)	测试用例个数	迭代次数比	时间比
J1	74.1	54.63	5.0	5.2	27.17	5	14.3	2.01
J2	410.5	97.60	12.2	24.6	43.39	10	16.7	2.25
J3	590.3	304.19	22.8	30.4	116.37	20	19.4	2.61
J4	2071.5	4624.36	30.9	81.2	3504.41	30	25.5	1.32
J5	14326.0	8640.16	22.1	668.3	5128.57	20	21.4	1.68
J6	550.0	804.64	6.7	123.2	499.89	6	4.5	1.61
J7	71.8	57764.81	6.9	13.4	8634.91	6	5.4	6.69
J8	5048.9	22240.35	20.9	170.3	8887.48	20	29.6	2.50
J9	67.5	391.56	5.3	6.1	325.89	5	11.1	1.20
J10	1033.8	359.61	9.4	20.6	233.61	9	50.2	1.54
J11	11086.7	37126.63	76.8	1825.6	22699.55	75	6.1	1.64
J12	9.5	69.06	8.3	2.2	52.64	7	4.3	1.31
J13	1042.6	193.20	20.5	286.9	150.56	20	3.6	1.28
Sum.	36383.2	132670.8	247.8	3258.0	50304.44	233	**Avg.** = 16.3	**Avg.** = 2.13

测试用例对弱变异测试有效性

ID	变异分支		GA		本文方法	
	总数	不可覆盖 变异分支个数	覆盖的 变异分支个数	覆盖率(%)	覆盖的 变异分支个数	覆盖率(%)
J1	111	0	111	100	111	100
J2	115	0	115	100	115	100
J3	213	0	213	100	213	100
J4	325	8	317	100	317	100
J5	314	15	299	100	299	100
J6	158	10	148	100	148	100
J7	377	29	348	100	348	100
J8	301	26	275	100	275	100
J9	805	127	656.8	96.87	657.0	96.90
J10	243	26	212.0	97.70	212.0	97.70
J11	912	132	768.8	98.56	769.9	98.71
J12	204	12	192	100	192	100
J13	242	5	237	100	237	100
Sum.	4320	390	3892.6	**Avg.** = 99.47	3893.9	**Avg.** = 99.49

测试用例对强变异测试有效性

ID	变异体		GA		本文方法	
	总数	等价变异体个数	杀死的变异体个数	变异得分(%)	杀死的变异体个数	变异得分(%)
J1	111	29	82.0	100	82.0	100
J2	115	18	95.9	98.87	96.4	99.38
J3	213	34	175.3	98.17	179.0	100
J4	325	40	276.5	97.02	277.7	97.44
J5	314	43	264.4	97.57	265.1	97.83
J6	158	12	146.0	100	146.0	100
J7	377	39	325.6	96.33	326.7	96.66
J8	301	36	256.7	96.87	255.4	96.38
J9	805	156	618.8	95.35	618.2	95.25
J10	243	38	197.8	96.49	198.1	96.63
J11	912	154	735.3	97.01	736.5	97.16
J12	204	15	186.3	98.57	186.7	98.78
J13	242	28	207.7	97.06	208.6	97.48
Sum.	4320	642	3568.3	**Avg.** = 97.64	3576.4	**Avg.** = 97.92

小结

- 建立了变异测试用例生成问题的数学模型，通过求解该模型，能够生成杀死所有变异体的测试用例
- 设计了采用集合进化方法求解上述模型时，适应度函数和进化算子

- 用于13个基准程序测试，并与传统遗传算法比较
  - ✓ 所提方法能降低测试用例生成成本
  - ✓ 生成的测试用例集基于弱变异和强变异测试准则，均能有效杀死变异体

项目支持

1. 2014.1-2017.12，基于占优度与集合进化的并行程序变异测试数据自动生成，国家自然科学基金
2. 2011.1-2013.12，基于等价关系的可测试性转化理论与方法及其在复杂软件进化测试中的应用，国家自然科学基金
3. 2012.7-2015.6，基于集合进化的复杂软件变异测试数据自动生成方法，江苏省自然科学基金
4. 2009.1-2011.12，基于不确定约束多目标进化优化理论的复杂软件测试数据自动生成，江苏省“六大人才高峰”高层次人才项目
5. 2010.1-2012.12，复杂软件多路径覆盖测试数据进化生成，江苏省“333高层次人才培养工程”项目
6. 2011.1-2013.12，基于等价关系的可测试性转化理论与应用，高等学校博士学科点专项科研基金

相关论文

1. Gong D. W., Tian T., Yao X. J., Grouping target paths for evolutionary generation of test data in parallel, The Journal of Systems and Software, 2012, 85(11): 2531-2540.
2. Gong D. W., Zhang W. Q., Yao X. J., Evolutionary generation of test data for many paths coverage based on grouping, The Journal of Systems and Software, 2011, 84(12): 2222-2233.
3. Gong D. W., Yao X. J., Automatic detection of infeasible paths in software testing, IET Software, 2010, 4(5): 361-370.
4. Gong D. W., Yao X. J., Testability transformation based on equivalence of target statements, Neural Computing & Applications, 2012, 21(8): 1871-1882.

相关论文

5. Tian T., Gong D. W.*, Test data generation for path coverage of message-passing parallel programs based on co-evolutionary genetic algorithms, Automated Software Engineering, 2014, DOI: 10.1007/s10515-014-0173-z.
6. Yao X. J., Gong D. W., Constrained multi-objective test data generation based on set evolution, IET Software.(Accepted)
7. 张功杰,巩敦卫,姚香娟, 基于变异分析和集合进化的测试用例生成方法, 计算机学报.(已录用)
8. 张功杰,巩敦卫,姚香娟, 基于统计占优分析的变异测试, 软件学报.(已录用)
9. …

主要内容

1 变异测试

2 进化测试

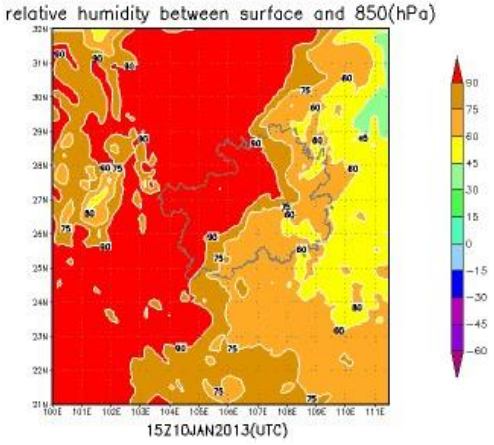
3 我们的工作

4 研究展望

并行程序

- 含有两个及以上并行执行进程（线程）的程序
- 以专用高性能并行机或普通集群系统作为硬件平台
- 能够大幅度缩短问题求解时间，提高求解精度
- 应用领域

天气预报

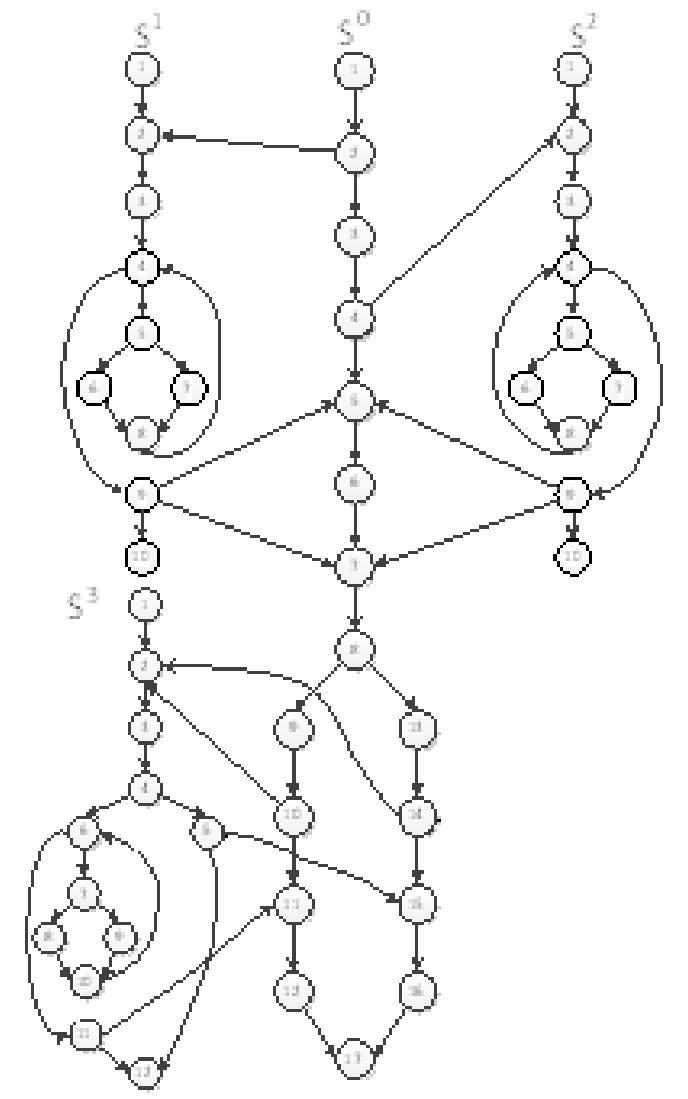


环境科学

药物设计

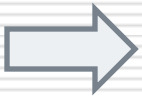
消息传递并行程序

- 结构特点
  - ✓ 具有传统串行程序的很多语句类型
  - ✓ 包含若干并行环境控制和通信语句
  - ✓ 程序的执行具有不确定性



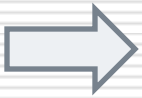
变异测试存在的问题

- 变异语句条数多
- 变异语句类型多
- 变异算子多



- 已有变异分支构建方法不再适用
- 已有占有关系不再适用
- 已有变异体约简方法不再适用

变异体急剧增多



并行程序变异测试非常具有挑战性

想法

- 通过变异算子，生成变异体
- 基于弱变异测试准则，生成对应的变异分支
- 将变异分支插入到原程序的合适位置，形成新的被测程序
- 基于占优度，约简变异体
- 采用进化优化方法，生成杀死变异体的测试用例

谢谢大家！