武漢大學
WUHAN UNIVERSITY


http://cstar.whu.edu.cn/

Nov. 17, 2018
CSBSE, Beijing

# 遗 传 配 置 采 样

## 可配置系统崩溃故障发现策略的学习方法

**玄跻峰**

武汉大学 计算机学院

智能化软件与服务研究所

jxuan@whu.edu.cn

高可配置系统无处不在。

# 研究背景

高可配置系统无处不在


*C++ Language Options*

See Section 3.5 [Options Controlling C++ Dialect], page 42.

```
-fabi-version=n -fno-access-control
-faligned-new=n -fargs-in-order=n -fcheck-new
-fconstexpr-depth=n -fconstexpr-loop-limit=n
-ffriend-injection
-fno-elide-constructors
-fno-enforce-eh-specs
-ffor-scope -fno-for-scope -fno-gnu-keywords
-fno-implicit-templates
-fno-implicit-inline-templates
-fno-implement-inlines -fms-extensions
-fnew-inheriting-ctors
-fnew-ttp-matching
-fno-nonansi-builtins -fnothrow-opt -fno-operator-names
-fno-optional-diags -fpermissive
-fno-pretty-templates
-frepo -fno-rtti -fsized-deallocation
-ftemplate-backtrace-limit=n
-ftemplate-depth=n
-fno-threadsafe-statics -fuse-cxa-atexit
-fno-weak -nostdinc++
-fvisibility-inlines-hidden
-fvisibility-ms-compat
-fext-numeric-literals
-Wabi=n -Wabi-tag -Wconversion-null -Wctor-dtor-privacy
-Wdelete-non-virtual-dtor -Wliteral-suffix -Wmultiple-inheritance
```

```
-faggressive-loop-optimizations -falign-functions[=n]
-falign-jumps[=n]
-falign-labels[=n] -falign-loops[=n]
-fassociative-math -fauto-profile -fauto-profile[=path]
-fauto-inc-dec -fbranch-probabilities
-fbranch-target-load-optimize -fbranch-target-load-optimize2
-fbtr-bb-exclusive -fcaller-saves
-fcombine-stack-adjustments -fconserve-stack
-fcompare-elim -fcprop-registers -fcrossjumping
-fcse-follow-jumps -fcse-skip-blocks -fcx-fortran-rules
-fcx-limited-range
-fdata-sections -fdce -fdelayed-branch
virtualize -fdevirtualize-speculatively
nsive-optimizations -ffat-lto-objects
float-store -fexcess-precision=style
=style -ffunction-sections
-las -fgcse-lm -fgraphite-identity
-fif-conversion
ing
ions-called-once -finline-limit=n
-fipa-cp-clone
e-const -fipa-reference -fipa-icf
ressure
e-save-slots
ence -fisolate-erroneous-paths-attribute
```

# 研究背景

高可配


*C++ Language Options*

See Section 3.5 [Options Controlling C++ Dialect], page 42.

```
-fabi-version=n -fno-access-control
-faligned-new=n -fargs-in-order=n -fcheck-new
-fconstexpr-depth=n -fconstexpr-loop-limit=n
-ffriend-injection
-fno-elide-constructors
-fno-enforce-eh-specs
-ffor-scope -fno-for-scope -fno-gnu-keywords
-fno-implicit-templates
-fno-implicit-inline-templates
-fno-implement-inlines -fms-extensions
-fnew-inheriting-ctors
-fnew-ttp-matching
-fno-nonansi-builtins -fnothrow-opt -fno-operator-names
-fno-optional-diags -fpermissive
-fno-pretty-templates
-frepo -fno-rtti -fsized-deallocation
-ftemplate-backtrace-limit=n
-ftemplate-depth=n
-fno-threadsafe-statics -fuse-cxa-atexit
-fno-weak -nostdinc++
-fvisibility-inlines-hidden
-fvisibility-ms-compat
-fext-numeric-literals
-Wabi=n -Wabi-tag -Wconversion-null -Wctor-dtor-privacy
-Wdelete-non-virtual-dtor -Wliteral-suffix -Wmultiple-inheritance
```

# 研究背景

高可配


配置项

配置项

配置项

. . .

系统配置

性能

功能

隐藏故障

*C++ Language Options*

See Section 3.5 [Options Controlling C++ Dialect], page 42.

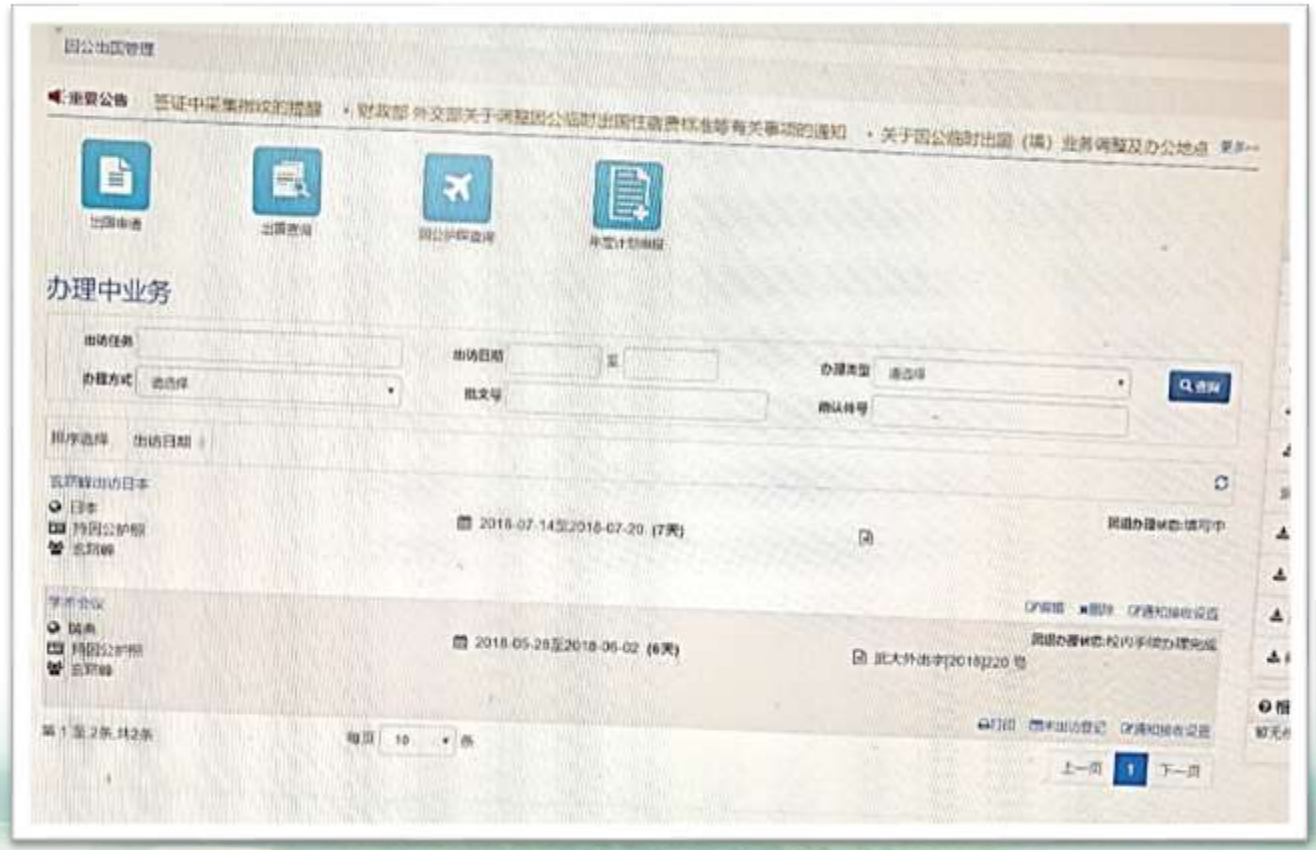
```
-fabi-version=n -fno-access-control
-faligned-new=n -fargs-in-order=n -fcheck-new
-fconstexpr-depth=n -fconstexpr-loop-limit=n
-ffriend-injection
-fno-elide-constructors
-fno-enforce-eh-specs
-ffor-scope -fno-for-scope -fno-gnu-keywords
-fno-implicit-templates
-fno-implicit-inline-templates
-fno-implement-inlines -fms-extensio
-fnew-inheriting-ctors
-fnew-ttp-matching
-fn -nonansi-builtins -fnothrow-opt -fno-operator-names
-fn
-fn
-fr
-ft
-ft
-fn -threadsafe-statics -fuse-cxa-at exit
-fno-weak -nostdinc++
-fvisibility-inlines-hidden
-fvisibility-ms-compat
-fext-numeric-literals
-Wabi=n -Wabi-tag -Wconversion-null -Wctor-dtor-privacy
-Wdelete-non-virtual-dtor -Wliteral-suffix -Wmultiple-inheritance
```

# CMOS – 古老的高可配置系统

# 常见的可配置系统

MIS - 国际交流部的因公护照访问管理系统

只有提交PDF文件时会激发无响应故障 – **文件类型配置项缺失**



高可配置系统无处不在。

系统配置故障也是。。。

# 可配置系统的那些配置

对于只包含3个配置项的系统，假设每个配置项只有布尔值：也就是 0 表示 disable，1 表示 enable

| 配置ID | 配置项A | 配置项B | 配置项C |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 0 | | |
| 4 | 0 | | |
| 5 | 1 | 0 | 0 |
| 6 | 1 | 0 | 1 |
| 7 | 1 | 1 | 0 |
| 8 | 1 | 1 | 1 |

**n个 配置项有2^n 个备选配置**

# 可配置系统的那些配置

对于只包含3个配置项的系统，假设每个配置项只有布尔值：也就是 0 表示 disable，1 表示 enable

| 配置ID | 配置项A | 配置项B | 配置项C |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 |
| 5 | 1 | 0 | 0 |
| 6 | 1 | 0 | 1 |
| 7 | 1 | 1 | 0 |
| 8 | 1 | 1 | 1 |

# 可配置系统的那些配置

如何检查全部配置项的全部配置（组合）？

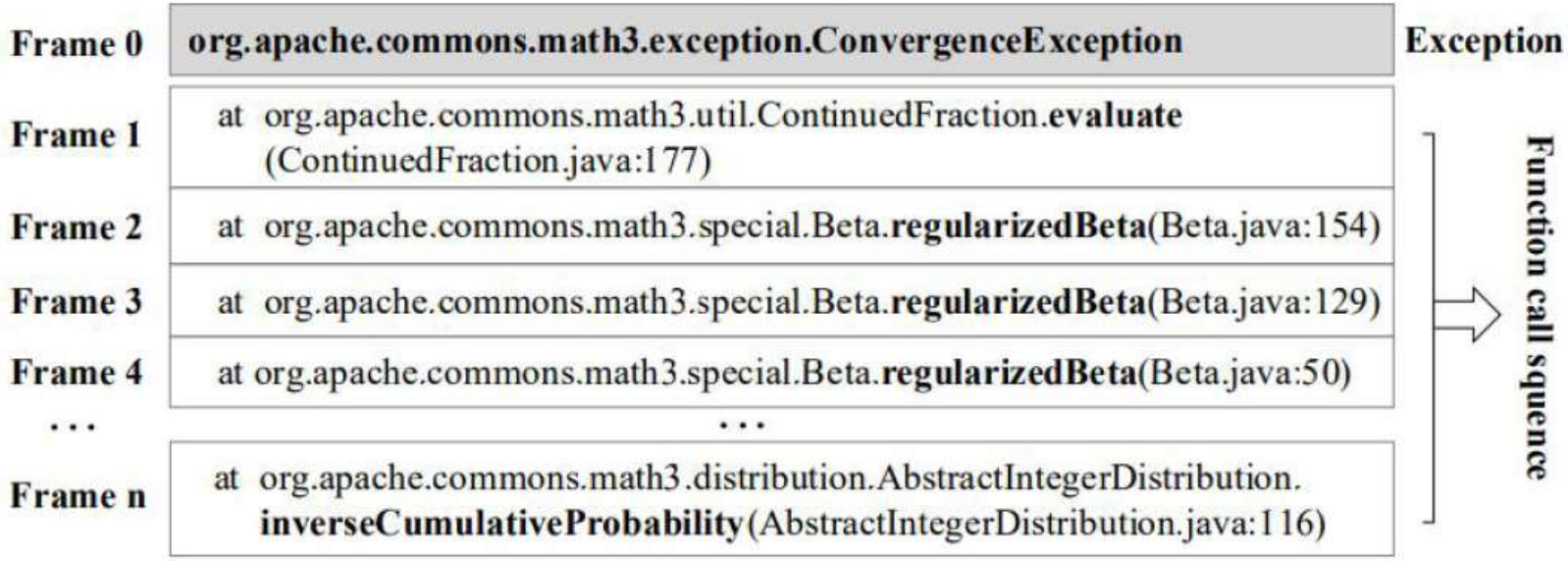
检查全部配置引发的故障

# 可配置系统的那些配置

**如何检查全部配置项的全部配置（组合）？**

**检查全部配置引发的故障**

**多数是系统崩溃**



# 可配置系统的那些配置

如何检查全部配置项的全部配置（组合）？

检查全部配置引发的故障

多数是系统崩溃

如何检查很少的配置但发现很多的潜在崩溃?

检查部分配置 - 系统配置采样

# 组合测试与采样策略

One-disabled策略

| ID | 配置项A | 配置项B | 配置项C |
|---|---|---|---|
| 1 | 0 | 1 | 1 |
| 2 | 1 | 0 | 1 |
| 3 | 1 | 1 | 0 |

One-enabled策略

| ID | 配置项A | 配置项B | 配置项C |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 |

# 组合测试与采样策略

2-wise策略

| ID | 配置项A | 配置项B | 配置项C |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 |
| 3 | 1 | 1 | 0 |
| 4 | 1 | 0 | 1 |

{A, B}, {A,C}, {B, C}

3-wise　4-wise　5-wise　6-wise

*t*-wise 策略

# 组合测试与采样策略

Random策略

| ID | 配置项A | 配置项B | 配置项C |
|---|---|---|---|
| 1 | 1 | 1 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 |
| 4 | 1 | 0 | 1 |

配置项A, B, C的随机选择

# 重新审视 采样策略

One-disabled策略

| ID | 配置项A | 配置项B | 配置项C |
|---|---|---|---|
| 1 | 0 | 1 | 1 |
| 2 | 1 | 0 | 1 |
| 3 | 1 | 1 | 0 |

two-wise策略

| ID | 配置项A | 配置项B | 配置项C |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 |
| 3 | 1 | 1 | 0 |
| 4 | 1 | 0 | 1 |

{A, B}, {A,C}, {B, C}

# 重新审视 采样策略

One-disabled策略

采样动作

| ID | 配置项A | 配置项B | 配置项C |
|---|---|---|---|
| 1 | | | |

重复
选择一个配置项
做disabled

two-wise策略

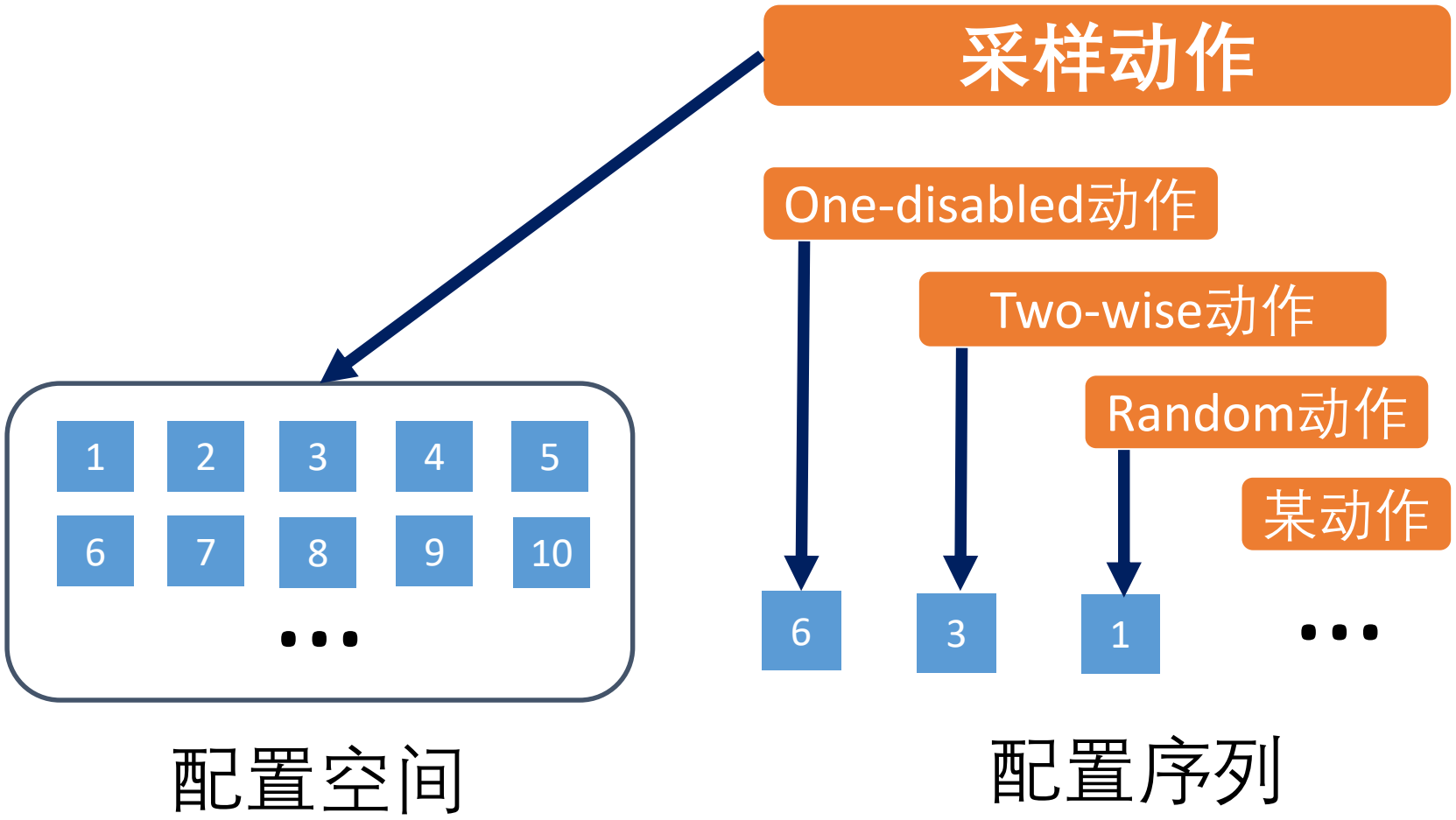
采样动作

| ID | 配置项A | 配置项B | 配置项C |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 4 | 1 | 0 | 1 |

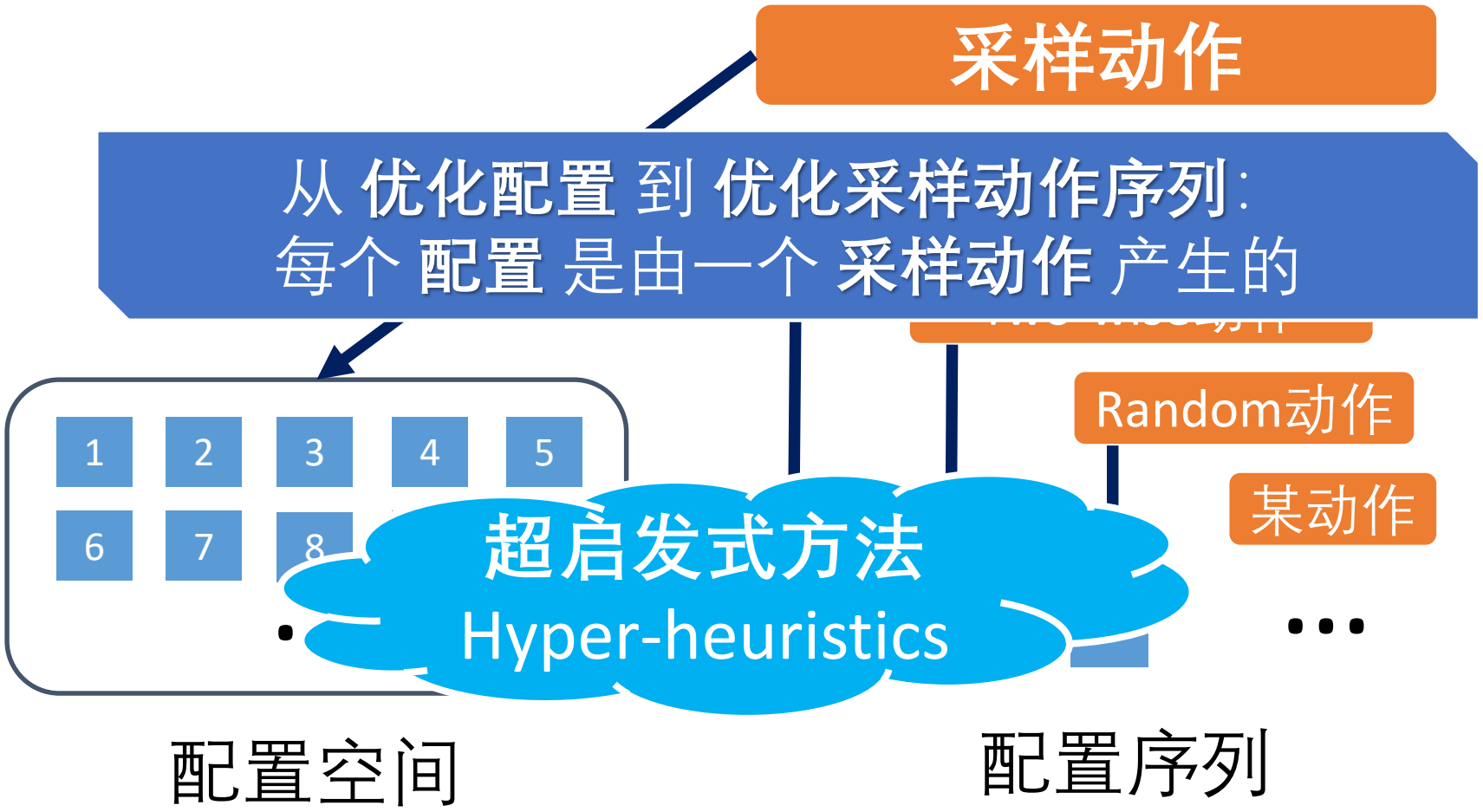
重复
选择两个配置项的
新值的组合

{A, B}, {A,C}, {B, C}

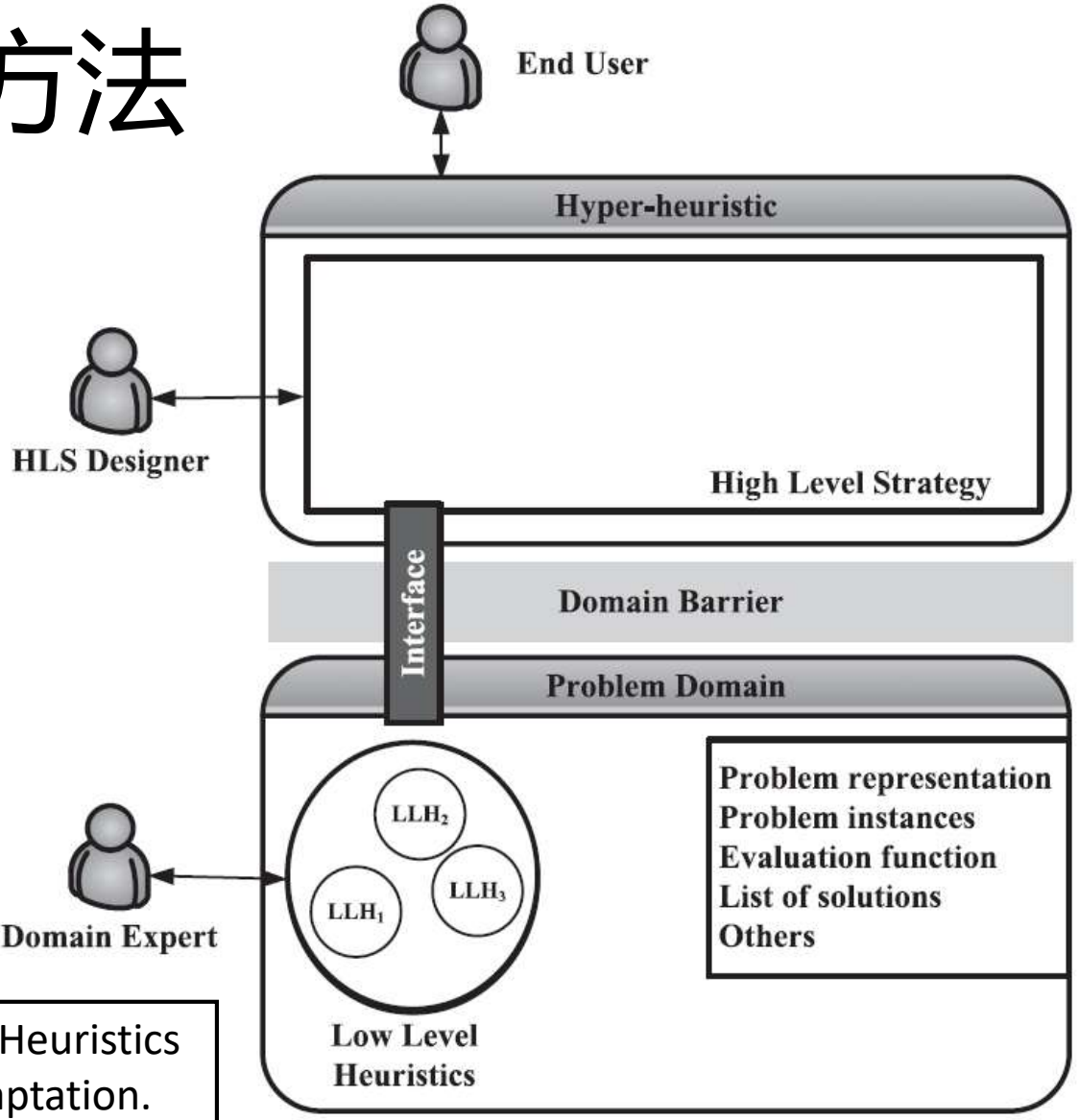
# 重新审视 采样策略


采样动作
One-disabled动作
Two-wise动作
Random动作
某动作
1
2
3
4
5
6
7
8
9
10
6
3
1
配置空间
配置序列

# 重新审视 采样策略


采样动作
从 优化配置 到 优化采样动作序列:
每个 配置 是由一个 采样动作 产生的
Random动作
某动作
1
2
3
4
5
6
7
8
超启发式方法
Hyper-heuristics
配置空间
配置序列

# 超启发式方法



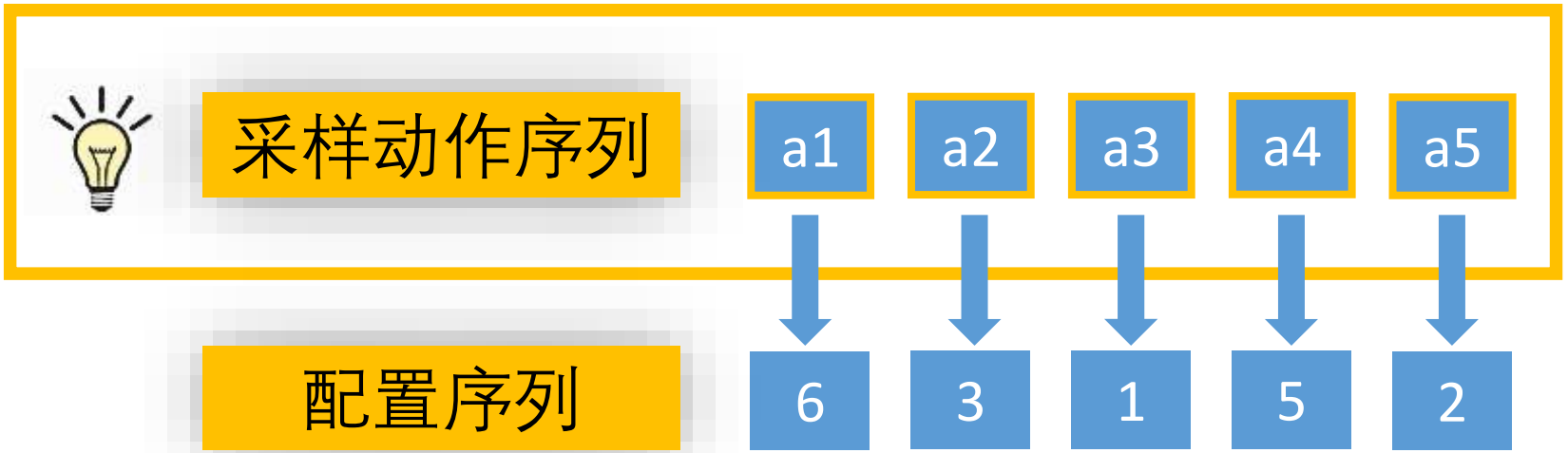

Ren, Jiang, **Xuan**, et al. Hyper-Heuristics with Low Level Parameter Adaptation. Evolutionary Computation 2012.

# 遗传配置采样 Genetic Configuration Sampling

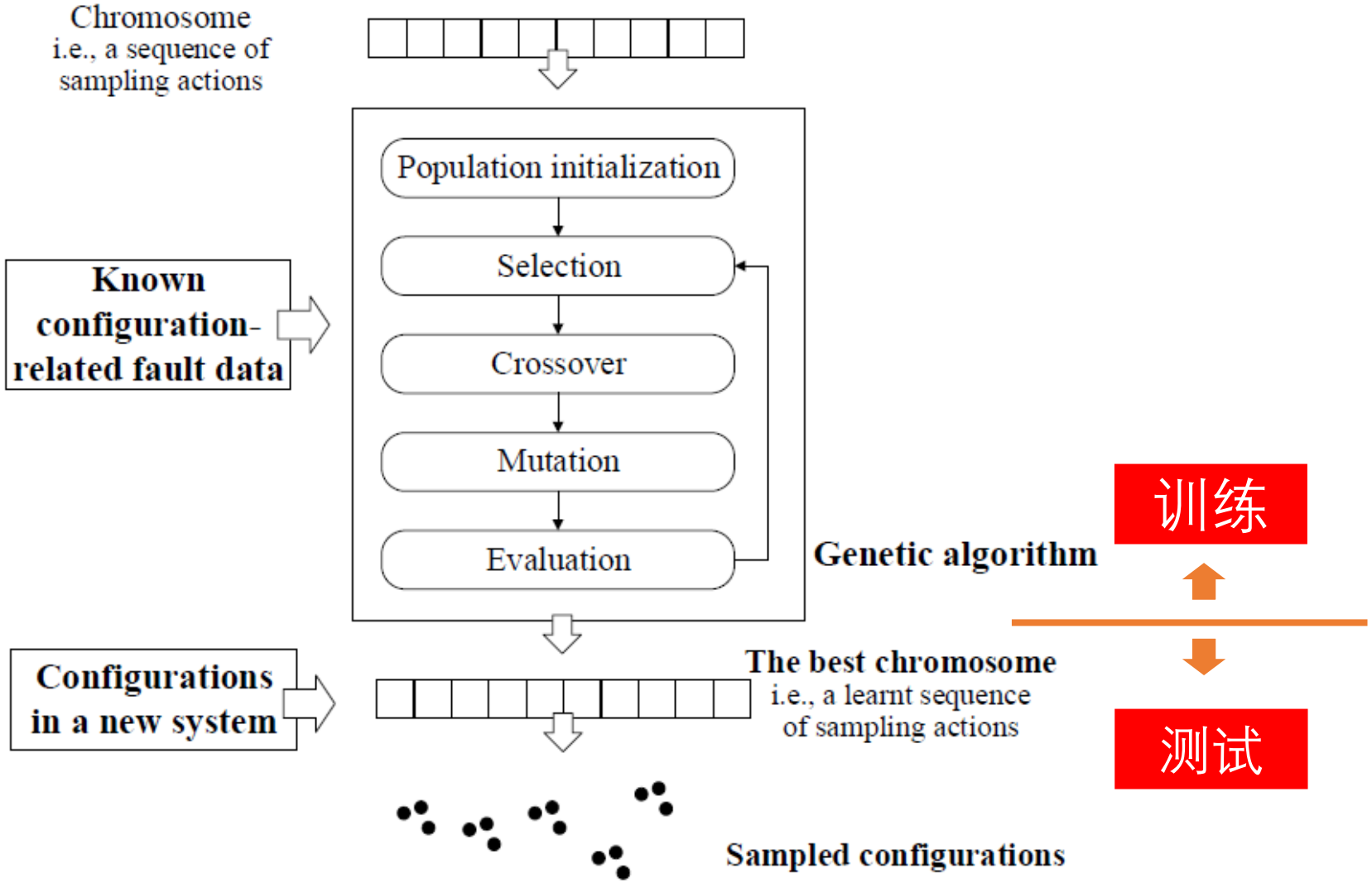
遗传提升
Genetic Improvement

# 遗传配置采样 Genetic Configuration Sampling

在哪儿使用遗传算法？



# 遗传配置采样




**Xuan**, Gu, et al. Genetic Configuration Sampling: Learning a Sampling Strategy for Fault Detection of Configurable Systems. GI@GECCO 2018.

# 遗传配置采样

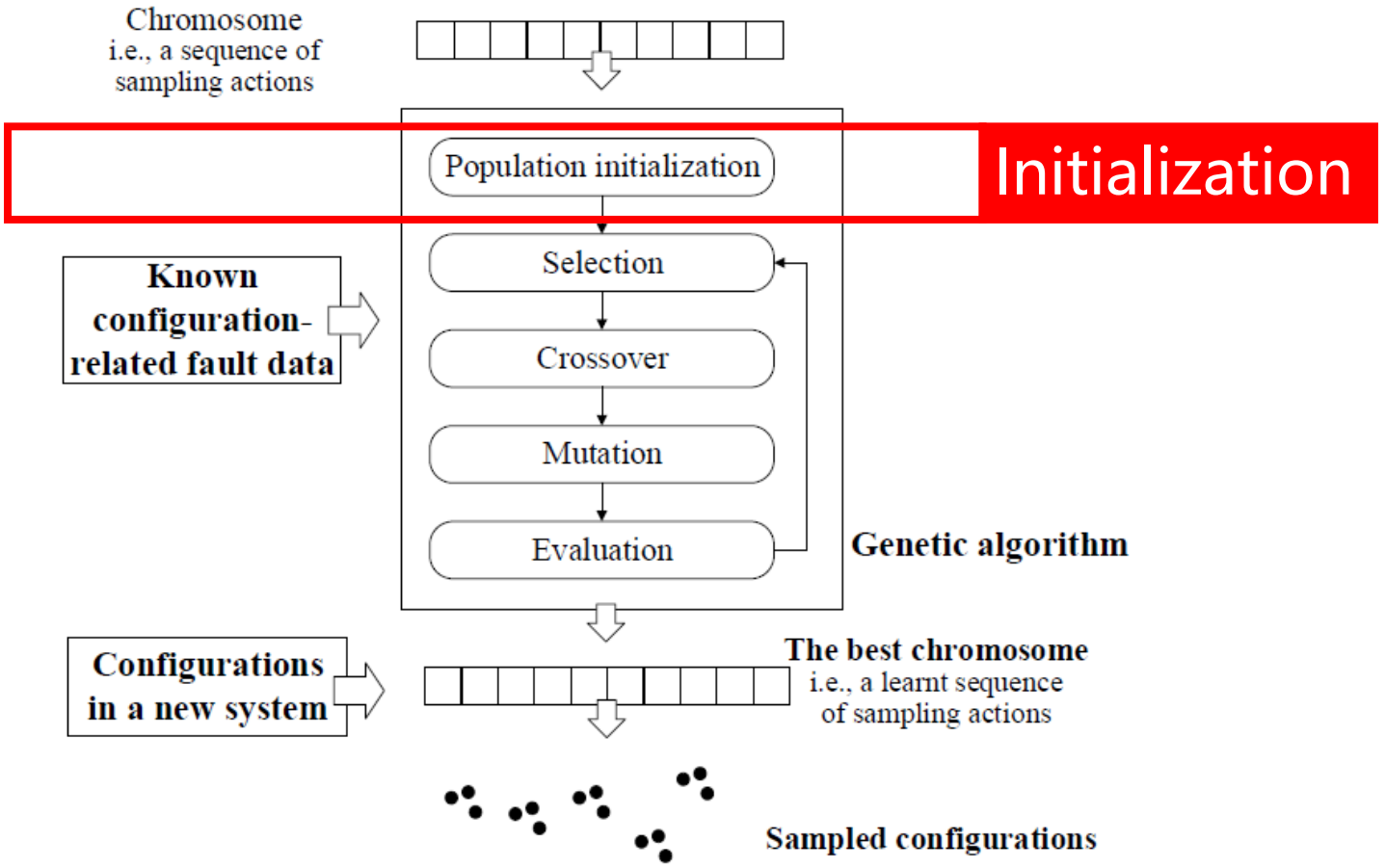
## Fitness function

$$fitness(chromosome) = \frac{1}{m}\sum_{i=1}^{m} undetected_i$$

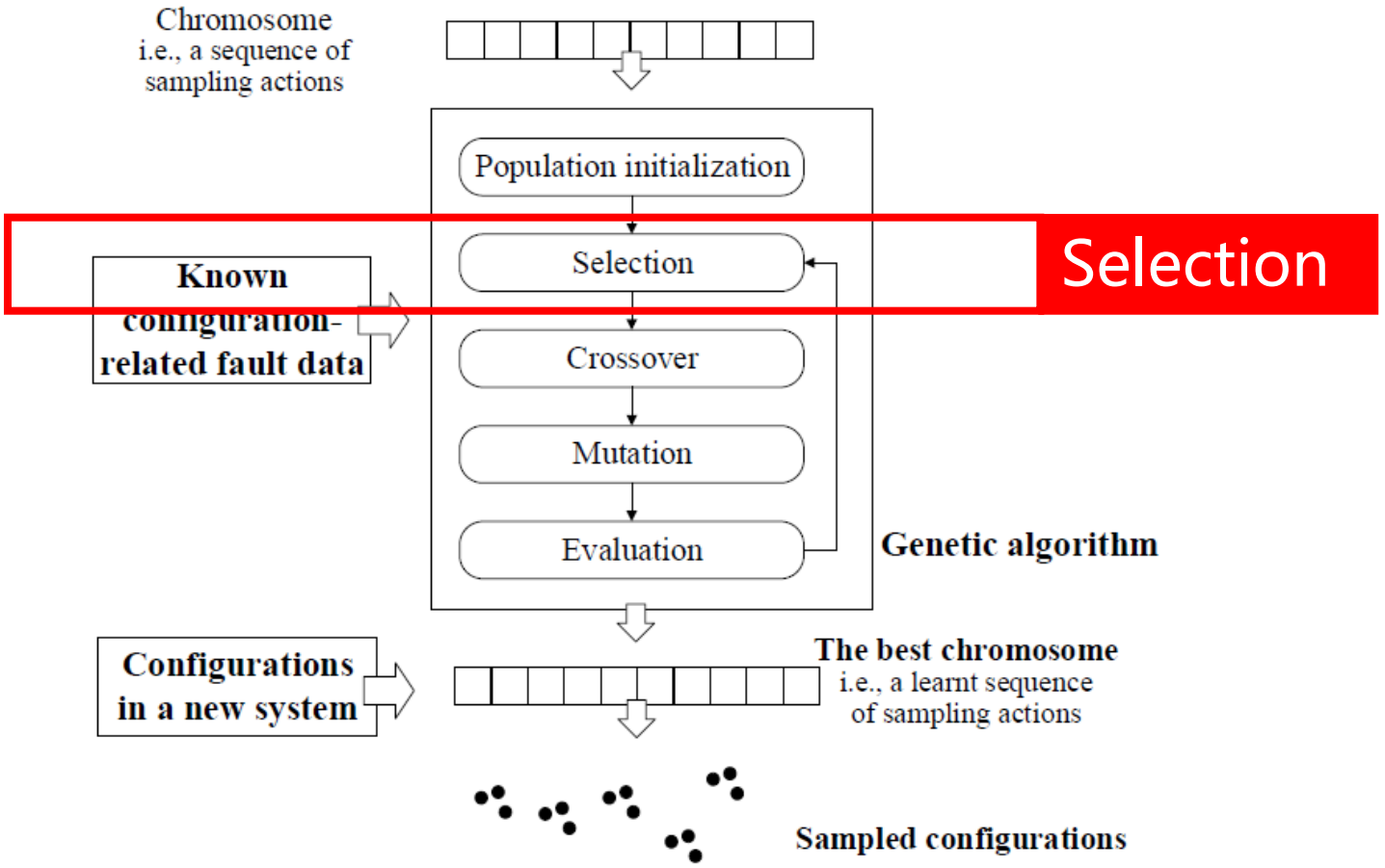
第 i 个采样序列中未发现的故障的个数

用每个染色体重复采样50次 (m=50)

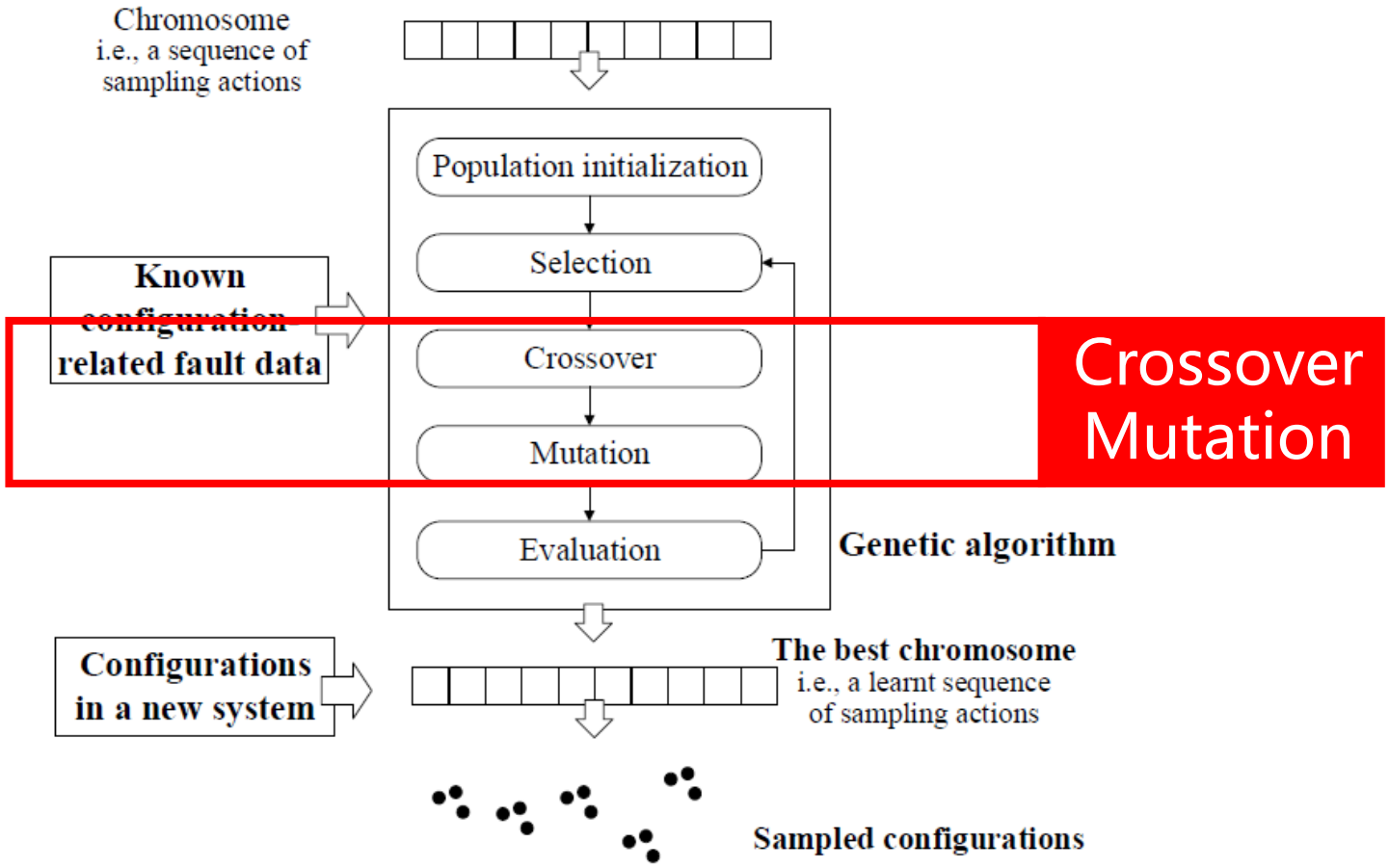
# 遗传配置采样


Chromosome
i.e., a sequence of
sampling actions
Population initialization
Initialization
Known
configuration-
related fault data
Selection
Crossover
Mutation
Evaluation
Genetic algorithm
Configurations
in a new system
The best chromosome
i.e., a learnt sequence
of sampling actions
Sampled configurations

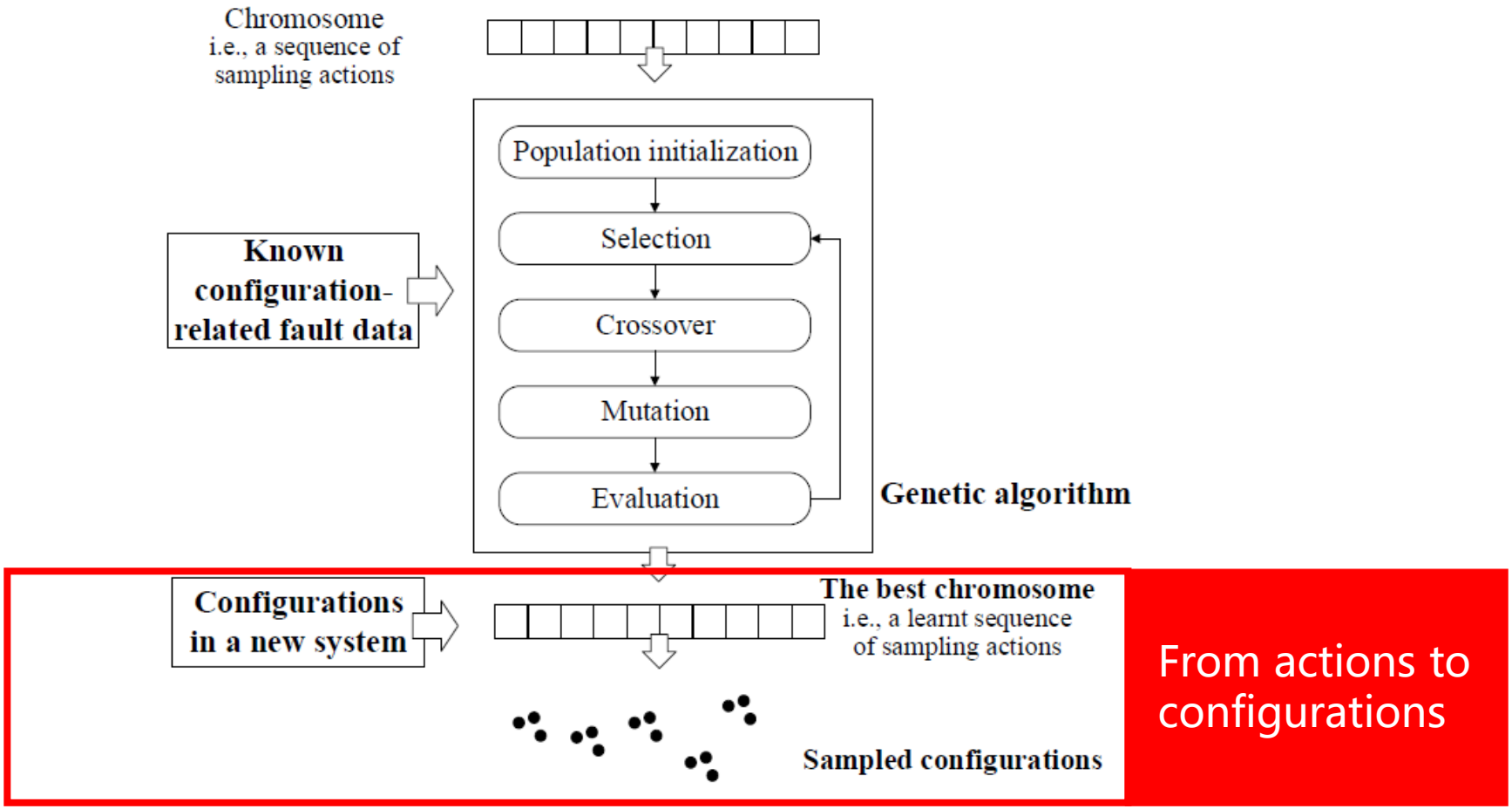
# 遗传配置采样


Chromosome
i.e., a sequence of
sampling actions
Population initialization
Selection
Crossover
Mutation
Evaluation
Genetic algorithm
Known
configuration-
related fault data
Selection
Configurations
in a new system
The best chromosome
i.e., a learnt sequence
of sampling actions
Sampled configurations

# 遗传配置采样


Chromosome
i.e., a sequence of
sampling actions
Population initialization
Selection
Known
configuration
related fault data
Crossover
Mutation
Crossover
Mutation
Evaluation
Genetic algorithm
Configurations
in a new system
The best chromosome
i.e., a learnt sequence
of sampling actions
Sampled configurations

# 遗传配置采样

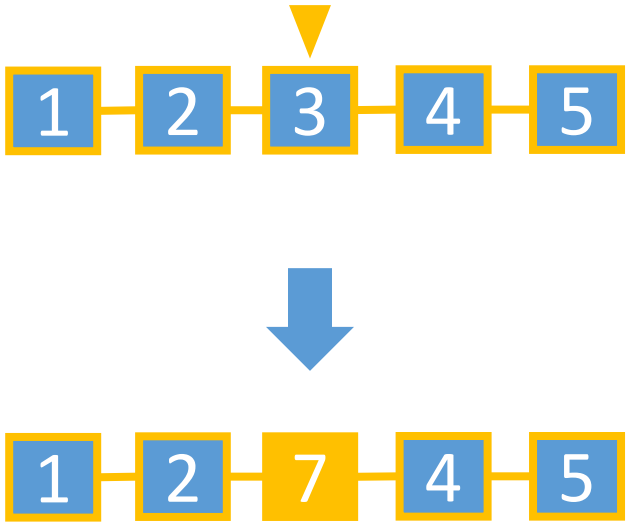

Chromosome
i.e., a sequence of
sampling actions
Population initialization
Selection
Crossover
Mutation
Evaluation
Genetic algorithm
Known
configuration-
related fault data
Configurations
in a new system
The best chromosome
i.e., a learnt sequence
of sampling actions
Sampled configurations
From actions to
configurations

# 遗传配置采样

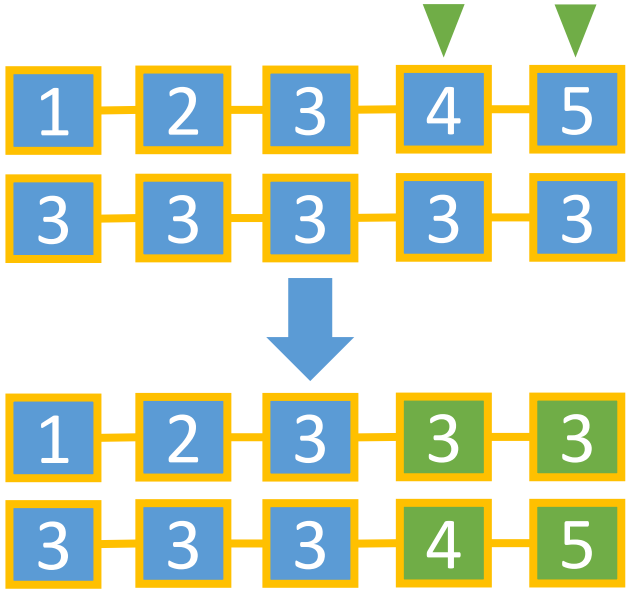

Population size= 20
Length of chromosome= 10
Repeating times during configurations=10

# 遗传配置采样

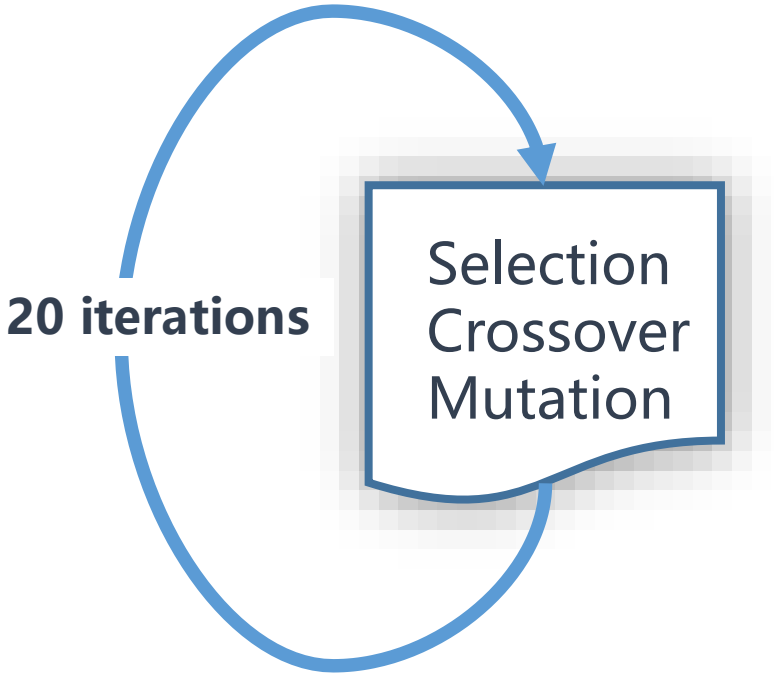
10% Mutation operator




90% Crossover operator



# 遗传配置采样





Optimized sequence of sampling actions

# 初步实验

Apache, BusyBox, Linux 数据集

**Table 1: Dataset of Three Real-World Systems, Apache, BusyBox and Linux**

| System | Domain | # Files | LoC | Configuration options | Faults |
|---|---|---|---|---|---|
| Apache | Web server application | 362 | 144,768 | 700 | 12 |
| BusyBox | Unix utility application | 805 | 189,722 | 1,418 | 10 |
| Linux | Operation system | 37,520 | 12,594,584 | 26,427 | 37 |

F. Medeiros, C. Kästner, M. Ribeiro, R. Gheyi, and S. Apel. A comparison of 10 sampling algorithms for configurable systems. ICSE 2016.

# 初步实验

Apache, BusyBox, Linux 自训练时得到的采样动作序列

**Table 2: Sequences of 10 Sample Actions that are Learnt from Three Systems**

| System | Sequence † |
|---|---|
| Apache | three-wise, one-enabled, five-wise, pair-wise, five-wise, four-wise, one-disabled, five-wise, pair-wise, six-wise |
| BusyBox | six-wise, four-wise, one-disabled, three-wise, four-wise, one-disabled, pair-wise, three-wise, one-enabled, one-disabled |
| Linux | pair-wise, six-wise, one-enabled, pair-wise, one-enabled, one-enabled, three-wise, three-wise, one-enabled, four-wise |

† We denote a sampling action with its sampling strategy, e.g., *three-wise* is short for an action of the *three-wise* strategy.

# 初步实验

## Apache 项目结果

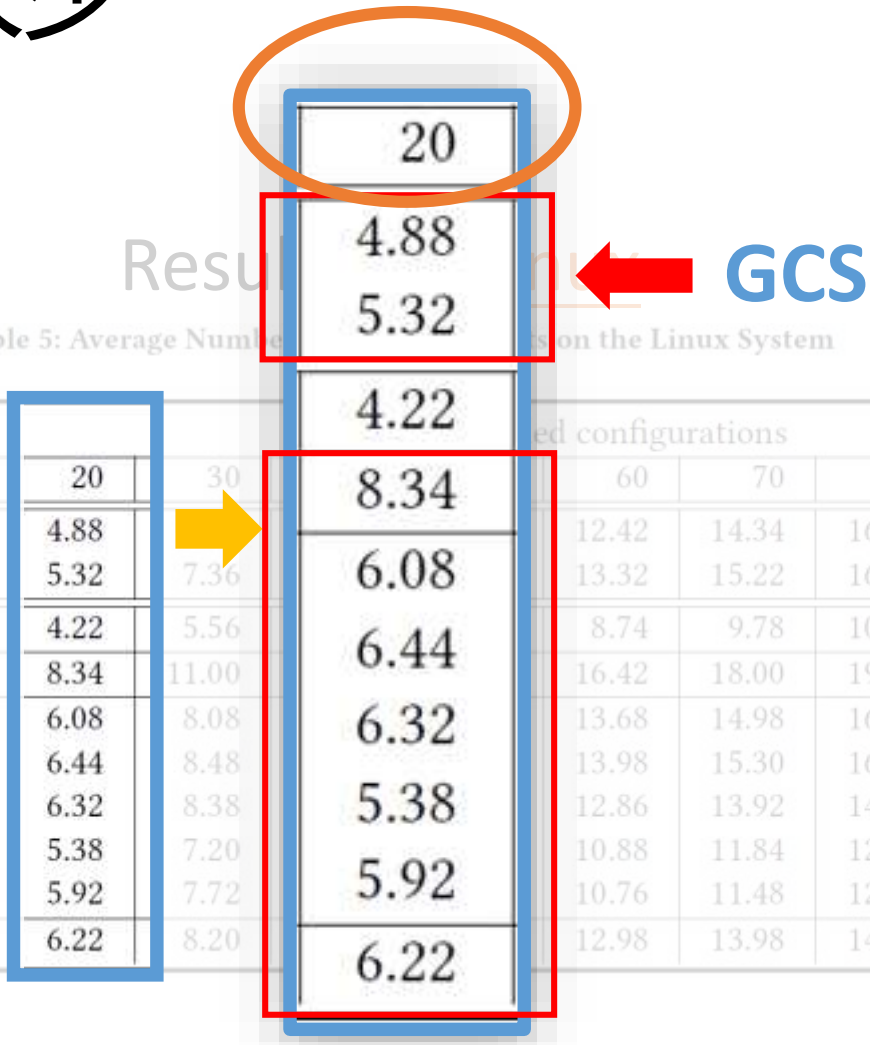
Table 3: Average Number of Detected Faults on the Apache System

| Sampling strategy | Size of sampled configurations | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| BusyBox⇒Apache | 3.88 | 6.64 | 8.68 | 10.34 | 11.08 | 11.34 | 11.46 | 11.58 | 11.72 | **11.80** |
| Linux⇒Apache | 4.28 | 7.20 | 9.32 | 10.74 | 11.02 | 11.20 | 11.26 | 11.32 | 11.46 | **11.54** |
| One-enabled | 1.32 | 2.44 | 3.52 | 4.24 | 5.28 | 6.44 | 7.52 | 8.22 | 9.00 | 9.70 |
| One-disabled | 6.48 | 7.84 | 8.64 | 9.16 | 9.60 | 9.84 | 10.18 | 10.44 | 10.62 | 10.80 |
| Pair-wise | 4.88 | 7.46 | 9.48 | 10.30 | 11.04 | 11.62 | - | - | - | - |
| Three-wise | 5.12 | 7.32 | 8.66 | 9.36 | 9.84 | 10.02 | 10.38 | 10.52 | 10.64 | 10.76 |
| Four-wise | 4.76 | 6.68 | 7.78 | 8.48 | 9.02 | 9.44 | 9.74 | 9.96 | 10.24 | 10.46 |
| Five-wise | 5.14 | 6.70 | 7.56 | 8.02 | 8.52 | 8.76 | 8.90 | 9.12 | 9.26 | 9.42 |
| Six-wise | 4.96 | 6.34 | 6.92 | 7.30 | 7.56 | 7.84 | 8.00 | 8.18 | 8.30 | 8.48 |
| Random | 4.98 | 7.22 | 8.42 | 9.12 | 9.58 | 9.92 | 10.18 | 10.40 | 10.50 | 10.68 |

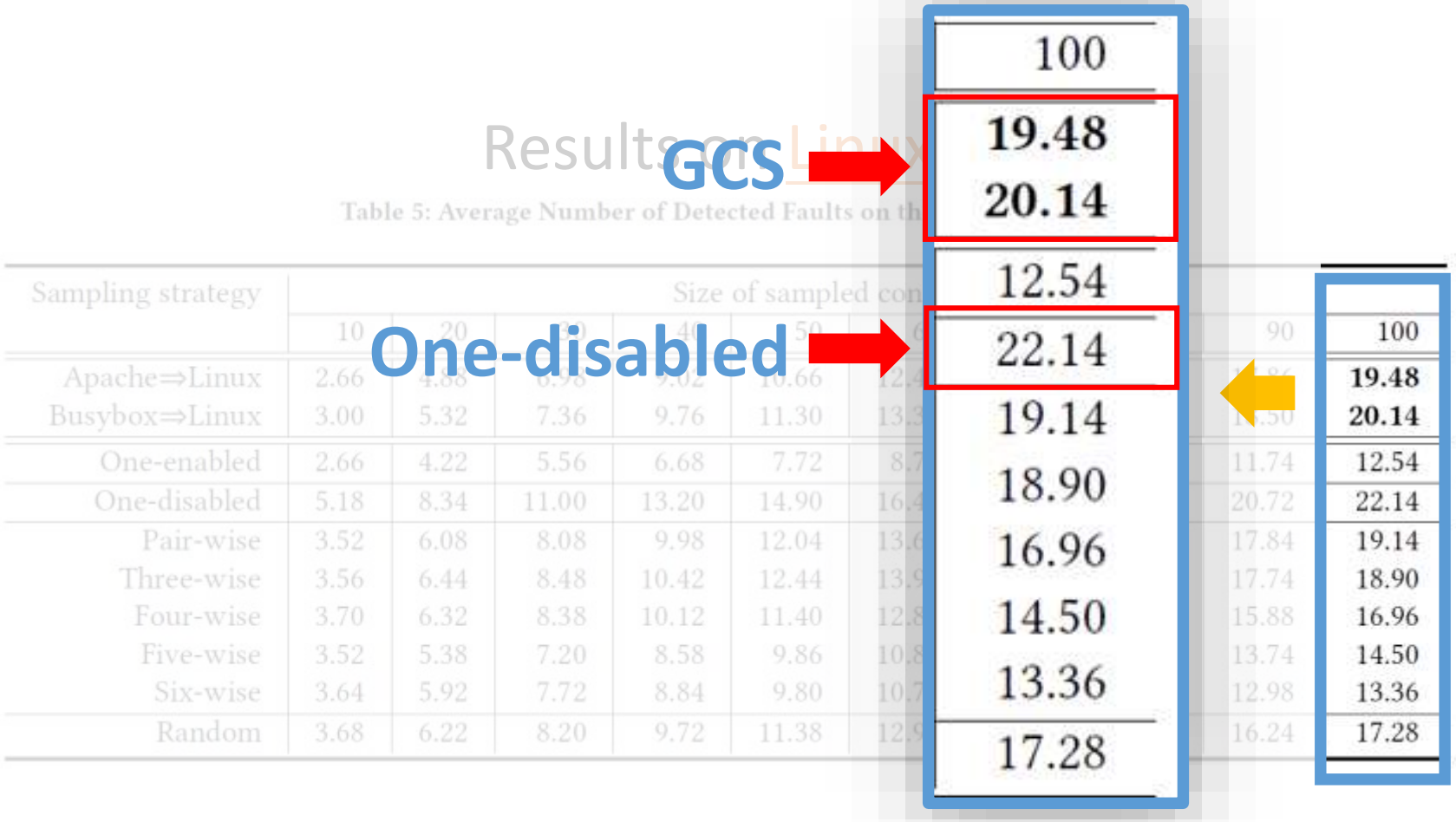
# 观察

- 遗传配置采样（GCS）在 Apache 和 Busybox 项目中能找到最多故障，在 Linux 中能找到第二多故障。
- GCS 采样数越多，越有机会找到更多的 fault。
- GCS 在每个项目上训练出的 sequence 作用于另一个项目的效果不同。

# 采样数太少



# 传统配置采样方法不稳定



# 讨论1

为什么 **遗传配置采样** 更好?

Hyper Heuristics

采样动作的搜索空间远比配置的搜索空间小。

# 讨论2

基于交叉项目训练的方案 稳定 吗?

Data distribution among projects ?

# 遗传配置采样

可配置系统崩溃故障发现策略的学习方法

除了未检查的系统配置，

还有什么会引发系统崩溃？

# 引发崩溃故障的场景重现

```
375 public void maybeConfigure(Project p, boolean configureChildren) {
376   if (proxyConfigured) {
377     return;
378   }
379   Object target = (wrappedObject i
            wrappedObject).getProxy()
380   // BUG: Object target could be n
381   IntrospectionHelper ih = Introspe
382   ...
```

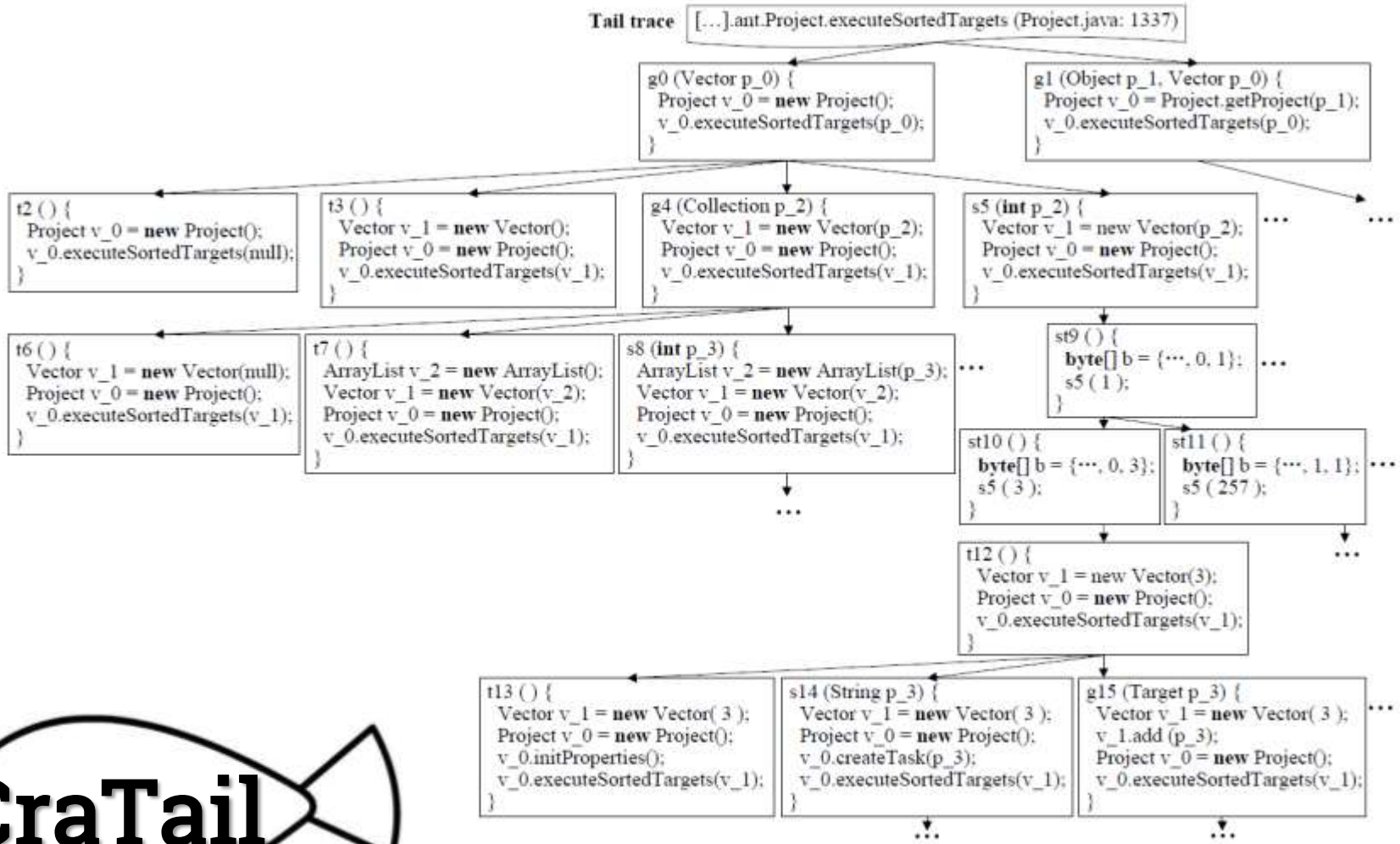
Bug-44689, Apache Ant 1.7.1

```
1 java.lang.NullPointerException:
2 [...].ant.RuntimeConfigurable.maybeConfigure(RuntimeConfigurable.java:381)
3 [...].ant.RuntimeConfigurable.maybeConfigure(RuntimeConfigurable.java:349)
4 [...].ant.Task.maybeConfigure(Task.java:202)
5 [...].ant.Task.perform(Task.java:347)
6 [...].ant.Target.execute(Target.java:357)
7 [...].ant.Target.performTasks(Target.java:385)
8 [...].ant.Project.executeSortedTargets(
```

```
1  public void tc__0() throws Throwable {
2    Project v_7 = new Project();
3    RuntimeConfigurable v_6 = new RuntimeC
4    ExecTask v_4 = new ExecTask();
5    v_4.setRuntimeConfigurableWrapper(v_6);
6    v_4.setProject(v_7);
7    Project v_3 = new Project();
8    Target v_2 = new Target();
9    v_2.addTask(v_4);
10   v_2.setProject(v_3);
11   Vector v_1 = new Vector();
12   v_1.add(v_2);
13   Project v_0 = new Project();
14   v_0.executeSortedTargets(v_1);
15 }
```

如何重现缺陷场景，辅助程序员修复程序？

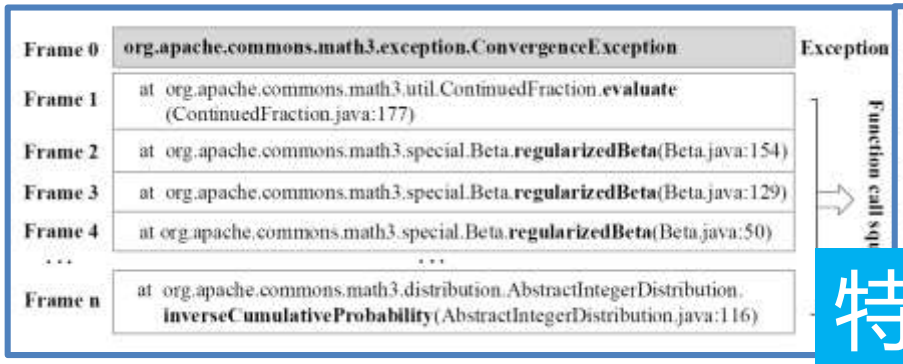
# 引发崩溃故障的场景重现




**CraTail**

# 引发崩溃故障的位置排查

## 如何刻画缺陷场景，用于度量程序修复补丁？




| Feature | Description |
|---|---|
| *Group ST – features related to the stack trace* | |
| ST01 | Type of the exception in the crash |
| ST02 | Number of frames of the stack trace |
| ST03 | Number of classes in the stack trace |
| ST04 | Number of methods in the stack trace |
| ST05 | Whether an overloaded method exists in the stack trace |
| ST06 | Length of the name in the top class |
| ST07 | Length of the name in the top function |

特征来自于 stack trace 和 source code 中，如：crash异常类型，报错函数基本信息，报错类基本信息等


| | | |
|---|---|---|
| CT16 | CB16 | Number of try blocks in the top/bottom function |
| CT17 | CB17 | Number of catch blocks in the top/bottom function |
| CT18 | CB18 | Number of finally blocks in the top/bottom function |
| CT19 | CB19 | Number of assignment statements in the top/bottom function |
| CT20 | CB20 | Number of method calls in the top/bottom function |
| CT21 | CB21 | Number of return statements in the top/bottom function |
| CT22 | CB22 | Number of unary operators in the top/bottom function |
| CT23 | CB23 | Number of binary operators in the top/bottom function |
| *Groups AT and AB – features normalized by LoC from Groups CT and CB* | | |
| AT01 | AB01 | CT08 / CT07 — CB08 / CB07 |

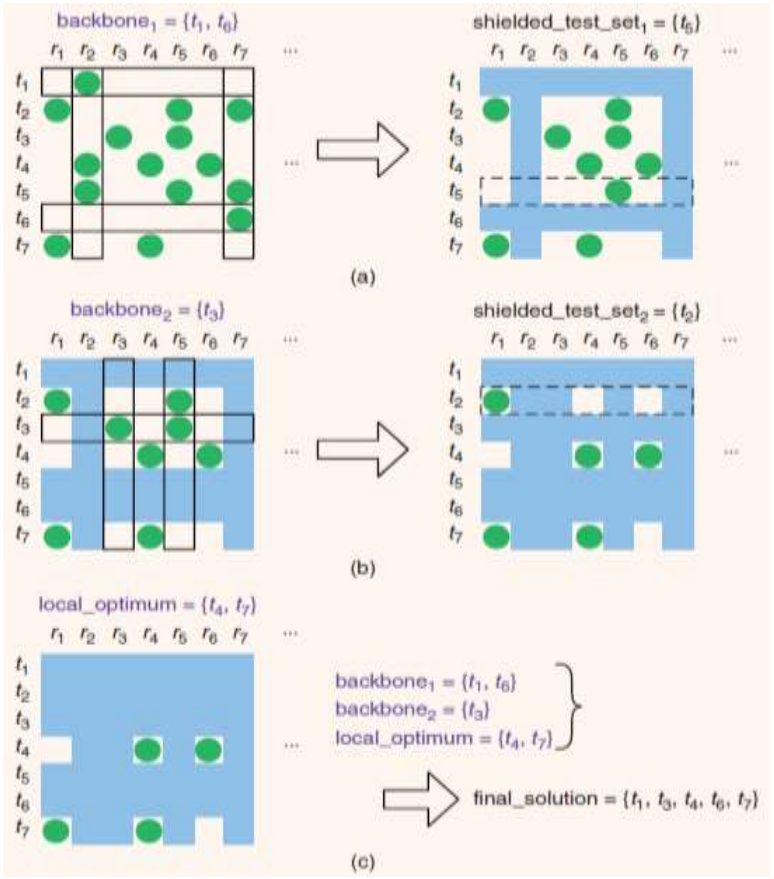
Gu, **Xuan**, et al. Does the Fault Reside in a Stack Trace? Assisting Crash Localization by Predicting Crashing Fault Residence. JSS 2018.

# 近期SBSE相关工作

# 基于逐层优化的测试集约简

## 减少测试用例数量

逐层规约测试用例，
优化测试用例数量的同
时保持代码覆盖程度

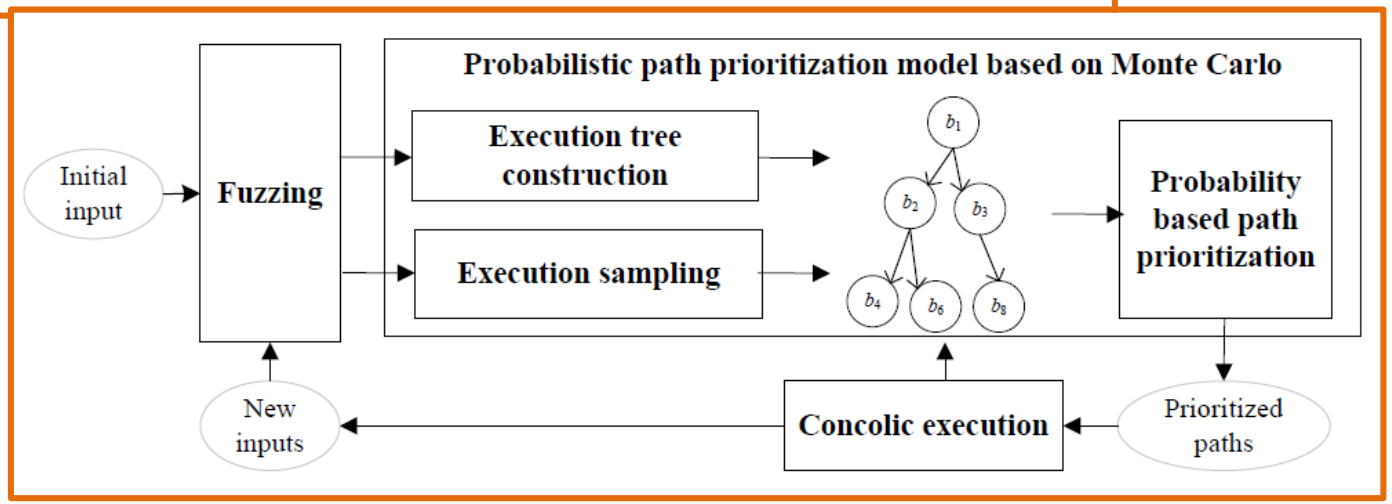




测试用例的逐层约简

Chi, **Xuan**, et al. Multi-Level Random Walk for Software Test Suite Reduction. IEEE Computational Intelligence Magazine, 2017

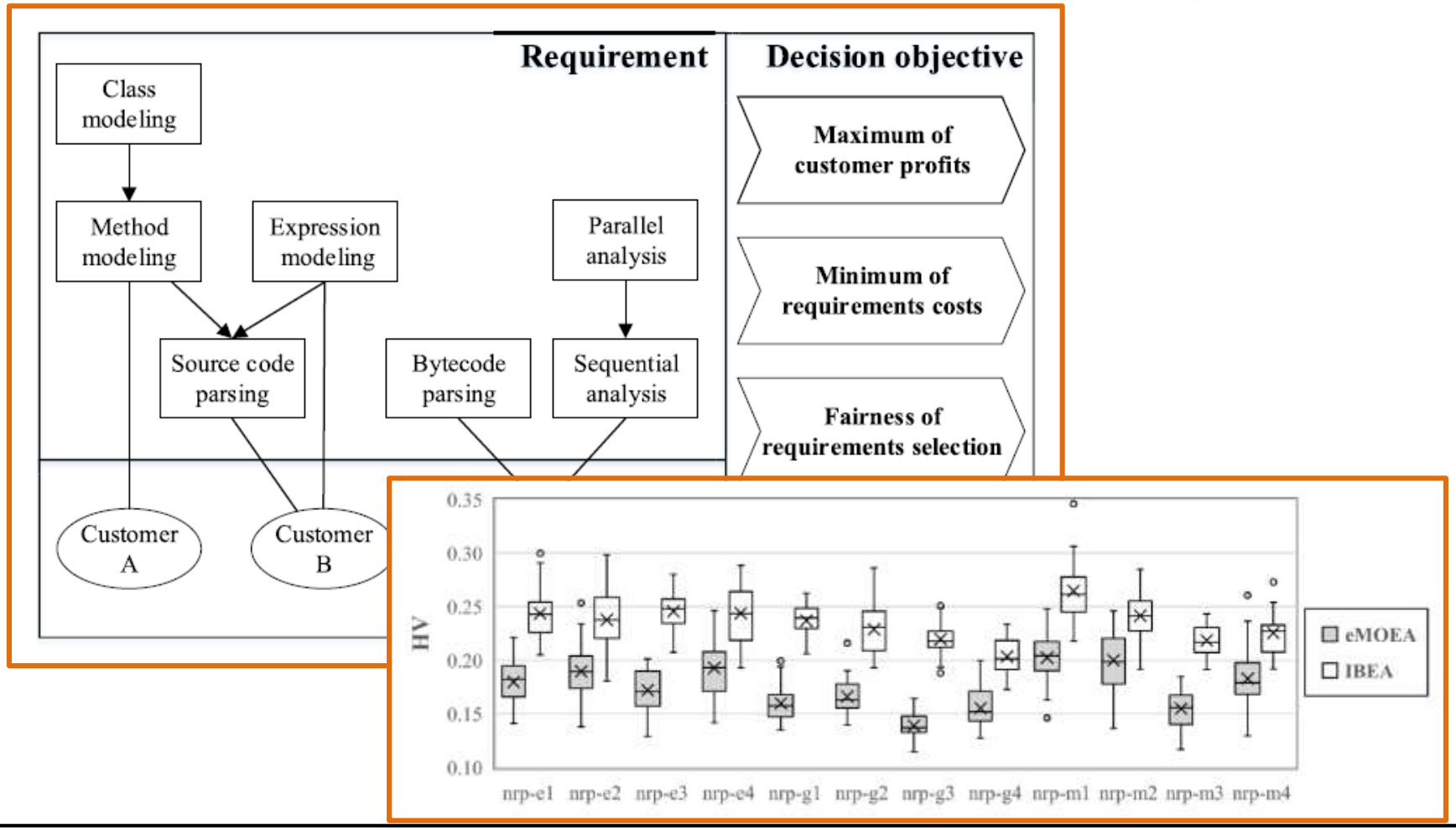
# 基于马尔科夫链的符号执行与模糊测试混合执行

```
     void main(argv) {                  int chk_in () {                  int is_valid(in) {
        switch (argv[1]) {      b6        revc(in)              b13       if all_char(in)
b1        case 'A':                       res = is_valid(in)    b14           return 1;
b2          chk_in();  break;   b7        if (!res)             b15       return 0;
b3        case 'B':             b8            return;                   }
b4          chk_in();  …        b9        cmd_id = is_cmd(in);          int is_cmd(in) {
            break;              b10       if (cmd_id == 1)      b16       if (strcmp(in, 'UPD') == 0)
b5        default: …            b11           //vulnerability   b17           return 1;
     }}                         b12       else …    }         b18       else return 2; }
```




Zhao, Duan, Yin, **Xuan**. Send Hardest Problems My Way: Probabilistic Path Prioritization for Hybrid Fuzzing. NDSS 2019.

# 软件需求选择问题的实证研究




Geng, Ying, Jia, Zhang, Liu, Guo, **Xuan**. Supporting Many-Objective Software Requirements Decision: An Exploratory Study on the Next Release Problem. Tech Report 2018.


CSTAR
首页
研究组简介
动态
团队成员
论文
软件
联系我们
English Version
武汉大学CSTAR软件测试分析研究组
主页 http://cstar.whu.edu.cn/
NASAC 2017
NASAC 2017
2017年11月3-5日 哈尔滨
ZHONG YANG STREET
HAGIA SOPHIA
TEAM MEMBER
TEAM MEMBER
TEAM MEMBER
欢迎老师和同学合作交流、学习讨论


NASAC 2018 缺陷修复Special Track

软件缺陷修复是近十年来的热点研究领域之一。从工业软件开发中的人工缺陷修复，到基于新型技术的自动缺陷修复，将会大量降低软件开发和维护的人力成本，快速提升软件质量。然而，目前自动的缺陷修复技术在实际应用中仍存在着极大的困难。

2018年11月24日（星期六）13:30--17:30

http://cstar.whu.edu.cn/nasac-repair18/

| 时间 | 主题 | 讲者 | 主持人 |
| --- | --- | --- | --- |
| 13:30-13:35 | 开幕 | | 熊英飞、玄跻峰 |
| 13:35-14:10 | **教程报告（Tutorials）1：**<br>缺陷修复技术介绍 | 熊英飞<br>北京大学 | 张成志 |
| 14:10-14:35 | **特邀学术报告 1：**<br>浮点计算精度缺陷的自动修复技术研究 | 毛晓光<br>国防科技大学 | |
| 14:35-15:00 | **特邀学术报告 2：**<br>Contract-base program repair without the contracts | 裴玉<br>香港理工大学 | |
| 15:00-15:20 | 茶歇 | | |
| 15:20-15:45 | **特邀学术报告 3：**<br>分而治之，<br>走向实用程序修复工具的一条可能途径 | 钟浩<br>上海交通大学 | 蔡彦 |
| 15:45-16:10 | **特邀学术报告 4：**<br>Repairing crashes in Android apps | 陈馨慧（Shin Hwei Tan）<br>南方科技大学 | |
| 16:10-16:35 | **特邀工业报告 1：**<br>面向 DevSecOps 的代码安全保障体系 | 董国伟<br>360 | 玄跻峰 |
| 16:35-17:00 | **特邀工业报告 2：**<br>阿里代码缺陷检测探索与实践 | 刘力华（息羽）<br>阿里 | |
| 17:00-17:25 | **特邀工业报告 3：**<br>代码自动修复：需求与收益 | 王千祥<br>华为 | |
| 17:25-17:30 | 闭幕 | | 熊英飞、玄跻峰 |

http://cstar.whu.edu.cn/nasac-repair18/

# 遗传配置采样
# Genetic Configuration Sampling

敬 请 批 评 指 正 !

玄跻峰

武汉大学

jxuan@whu.edu.cn