

# Manual for SMAC version v2.08.00-master

Frank Hutter & Steve Ramage

英国哥伦比亚大学计算机科学大学

Vancouver, BC V6T 1Z4, Canada

{hutter, seramage}@cs.ubc.ca

August 4, 2014

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	License . . . . .	3
1.2	System Requirements . . . . .	3
1.3	Version . . . . .	4
<b>2</b>	<b>Differences Between SMAC and ParamILS</b>	<b>4</b>
<b>3</b>	<b>Commonly Used Options</b>	<b>5</b>
3.1	Running SMAC . . . . .	5
3.2	Testing the Wrapper . . . . .	5
3.3	Verifying the Scenario . . . . .	5
3.4	Wall-Clock Limit . . . . .	6
3.5	Change Initial Incumbent . . . . .	6
3.6	State Restoration . . . . .	6
3.7	Warm-Starting the Model . . . . .	6
3.8	Named Rungroups . . . . .	7
3.9	More Options . . . . .	7
3.10	Shared Model Mode (Experimental) . . . . .	7
3.11	Offline Validation . . . . .	9
	3.11.1 限制验证运行中使用的实例数。 . . . . 9.	
	3.11.2 Disabling Validation . . . . .	9
	3.11.3 Standalone Validation . . . . .	9
<b>4</b>	<b>File Format Reference</b>	<b>10</b>
4.1	Option Files . . . . .	10
	4.1.1 Scenario File . . . . .	10
4.2	Instance File Format . . . . .	11
4.3	Feature File Format . . . . .	12
	4.4 参数配置空间格式。 . . . . 12.	

4.4.1	Parameter Declaration Clauses . . . . .	12
4.4.2	Conditional Parameter Clauses . . . . .	14
4.4.3	Forbidden Parameter Clauses . . . . .	15
<b>5</b>	<b>Wrappers</b>	<b>15</b>
5.1	Algorithm executable / wrapper . . . . .	15
5.1.1	Invocation . . . . .	15
5.1.2	Output . . . . .	17
5.2	Wrapper Output Semantics . . . . .	19
5.3	Wrappers & Native Libraries . . . . .	19
<b>6</b>	<b>Interpreting SMAC's Output</b>	<b>20</b>
6.1	Logging Output . . . . .	20
6.1.1	解释日志文件。 . . . . 21.	
6.2	State Files . . . . .	22
6.3	Trajectory File . . . . .	23
6.4	Validation Output . . . . .	23
6.5	JSON Output . . . . .	24
<b>7</b>	<b>Developer Reference</b>	<b>24</b>
7.1	Design Overview . . . . .	24
7.2	Class Overview . . . . .	25
7.3	算法执行和抽象工具包。 . . . . 27.	
7.4	Running SMAC in Eclipse . . . . .	27
<b>8</b>	<b>Acknowledgements</b>	<b>28</b>
<b>9</b>	<b>References</b>	<b>28</b>
<b>10</b>	<b>Appendix</b>	<b>29</b>
10.1	Return Codes . . . . .	29
10.2	java smac的版本历史记录。 . . . . 29.	
10.3	Known Issues . . . . .	33
10.4	Basic Options Reference . . . . .	35
10.4.1	SMAC Options . . . . .	35
10.4.2	Scenario Options . . . . .	35
10.4.3	方案配置限制选项。 . . . . 36.	
10.4.4	Algorithm Execution Options . . . . .	36
10.5	Complete Options Reference . . . . .	38
10.5.1	SMAC Options . . . . .	38
10.5.2	Random Forest Options . . . . .	44
10.5.3	Scenario Options . . . . .	46
10.5.4	场景配置限制选项。 . . . . 48.	
10.5.5	Algorithm Execution Options . . . . .	49
10.5.6	目标算法评估器选项。 . . . . 50.	
10.5.7	变换目标算法评估器装饰器选项。 . . . . 55.	

10.5.8分叉目标算法评估器装饰器选项。 . . . . .	57.
<b>10.5.9 Validation Options . . . . .</b>	<b>58</b>
10.5.10分析目标算法评估器选项。 . . . . .	60.
10.5.11黑洞目标算法评估器选项。 . . . . .	60.
10.5.12命令行目标算法评估器选项。 . . . . .	61.
10.5.13常量目标算法评估器选项。 . . . . .	63.
10.5.14进程间通信目标算法评估器选项。 . . . . .	63.
10.5.15预加载响应目标算法评估器。 . . . . .	65.
10.5.16随机目标算法评估器选项。 . . . . .	65.

# 1 Introduction

本文档是SMAC [2]的手册（连续模型的算法配置的首字母缩写）。SMAC旨在解决以下算法配置问题：给定参数化算法A的二进制文件，问题是解决问题的一组实例S，以及性能度量M，找到跨S优化m的参数设置。

稍微详细介绍，SMAC的用户必须提供：

参数算法A（从命令行调用的可执行文件），

参数 $\theta_1$ 的描述。 . . . ,  $\theta_n$ 及其域 $\theta_1$ , . . . ,  $\theta_n$ ,

一组基准实例， $\pi$ 和

目标函数测量和聚合算法预屈曲结果。

然后，SMAC以不同的参数配置执行算法A（参数的组合

$\theta_1, \dots, \theta_n$   $\theta_1 \times \dots \times \theta_n$ 在实例 $\pi \in \pi$ 上），搜索配置在所提供的目标下的基准实例中产生整体最佳性能的配置。有关详细信息，请参阅[2]；如果您在研究中使用SMAC，请引用该文章。如果您向我们发送了一封电子邮件，它也会很好 - 我们总是对其他应用领域感兴趣。

## 1.1 License

SMAC将在双重使用许可证下发布。免费提供学术和非商业用法。请联系我们讨论商业用法。

## 1.2 System Requirements

SMAC本身只需要Java 7或Newer来运行。

SMAC主要用于在UNIX上运行，如平台，但现在包括启动脚本，以便它可以在Windows上运行。在下面的所有示例中，您应该将.bat添加到每个可执行文件的末尾，例如./smac -scenario-file scen.txt -seed 1变为smac.bat - scenario-file scen.txt --seed 1。

大多数包括的方案（in ./表达方案/需要Ruby和Linux 32位li-野人跑。 ./表现方案/分析/使用的方案优化SMAC内部的功能，并且是完全跨平台的。目前有一种用于目前可用的Windows的场景\saps\saps-scenario-windows.txt。

## 1.3 Version

此版本的手册用于SMAC V2.08.00-Master-731。

Project	Version	Commit	Dirty Flag
aeatk	v2.08.00-master-766	85fc099c674a3d2cab86870ff0f731cb3b01c1e9	0
smac	v2.08.00-master-731	0e43c26c3d1fff66f2038054c98abd897e9949d1	0

注意：对于非主站构建，这些提交可能不包含构建中的所有内容。（即，无主构建可以使用未提交的更改构建）。如果脏标志为0，则表示提交包含此精确副本，1表示有一些未提交的更改，还有其他意味着在我们尝试生成此时发生了一些其他错误。

## 2 Differences Between SMAC and ParamILS

SMAC和Paramils之间存在许多差异，包括以下内容。

支持连续参数：虽然参数仅限于分类参数，但SMAC也是本身处理连续和整数的参数。有关详细信息，请参见第4.4.1节。

运行目标：此时并非所有Paramils的运行目标都得到支持。如果你需要一个不支持的目标请告诉我们。

实例的顺序：与参数相比，实例文件中的实例顺序无关紧要

配置时间预算和运行时开销：Paramils和SMAC都接受时间

预算作为输入参数。Paramils仅跟踪CPU时间目标算法报告并一旦这些运行时的总和超过时间预算;由于例如，它不会考虑开销。命令行呼叫目标算法。在每个目标算法运行的报告的CPU时间非常小（例如毫秒），这些未计算的开销实际上可以占主导地位的壁钟时间。SMAC通过选项 - 使用 -pu-time-in-tunertime和-wallclock-limit提供更灵活的运行时开销管理。有关挂钟时间限制的详细信息，请参见第3.4节。

恢复以前的运行：虽然在Paramils中是不可能的，但在SMAC中，您可以恢复以前从已保存的状态运行。请参阅第3.6节，了解如何使用状态恢复功能。第6.2节介绍了保存状态的文件格式。

功能文件：SMAC接受作为可选输入，提供有关的其他信息  
培训集中的实例;请参见第4.3节。

算法包装器：包装器语法已以SMAC扩展，以支持其他结果

“解决”领域。具体地，存在新的结果中止信号，即配置过程应该被中止（例如，因为包装器处于不一致状态，不应达到）。如果设置了Option-Abort-on-Run-Crash，并且第一个运行返回崩溃，则触发类似的行为。此外，包装器还可以将附加数据返回与运行1.有关更多信息的SMAC，有关更多信息，请参见第5.1.2节。

---

<sup>1</sup>数据将保存在状态保存中使用的运行和结果文件（第6.2节）中

实例文件与实例/种子文件：实例文件参数现在自动检测文件

符合Paramils的实例文件或实例种子文件格式。SMAC将后一种选择视为前者的别名。有关详细信息，请参阅第4.2节。虽然SMAC向后兼容以前（分开的）文件，但现在首选格式.CSV。

## 3 Commonly Used Options

### 3.1 Running SMAC

要开始使用现有的配置方案，您只需执行SMAC，如下所示：

```
./smac --scenario-file <file> --seed 1
```

这将使用文件中指定的方案上的默认选项执行SMAC。一些常见 -

本节中介绍了SMAC的使用非默认选项。--seed参数控制输出文件的种子和名称（支持并行独立运行）。--Seed-Offset参数允许您将输出文件夹名称保持简单，同时改变SMAC的实际种子。种子参数也是可选的，如果未设置，将自动选择。

### 3.2 Testing the Wrapper

SMAC包括通过algotest实用程序测试算法执行的方法。它需要所需的方案选项2

For example:

```
./algotest --scenario-file <scenario> --instance <instance>  
--config <config string> -P[name]=[value] -P[name]=[value]...
```

一些参数值得特别提及：

1. Config String语法是一个字符串，其中包含“-name =”值'“...您也可以随机指定  
这将生成随机配置或默认值，这将生成默认配置。
2. -p参数是可选的，允许在配置中覆盖特定值（这很有用  
主要用于随机且默认，允许您设置某些值）。要将sortalgo参数设置为合并，您将指定psortalgo =合并。

### 3.3 Verifying the Scenario

SMAC包含一个允许您测试方案的实用程序。它目前是Beta，但有比SMAC更多的理智检查通常会这样做。

For example:

```
./verify-scenario --scenarios ./scenarios/*.txt --verify-instances true
```

然而，该实用程序有一些限制：

- 1.它目前没有检查测试实例
- 2.方案文件可以在SMAC中指定非方案选项（以及一些示例方案实际上  
do），此实用程序不了解它们，并将报告错误。

---

2不幸的是它无法读取目前的方案文件

### 3.4 Wall-Clock Limit

```
./smac --scenario-file <file> --wallclock-limit <seconds> --seed 1
```

SMAC提供了在使用给定的挂钟时间后终止的选项。此选项可用于限制启动目标算法的开销运行，否则是未占用的。此选项不会覆盖`--tunertime-limit`或限制配置运行持续时间或其他选项；首先触及终止标准的任何终止标准。

### 3.5 Change Initial Incumbent

```
./smac --scenario-file <file> --initial-incumbent <config string>
```

SMAC提供指定初始现任的选项，默认情况下使用默认配置在参数文件中指定。“论－现任”的论点遵循与第3.2节相同的约定。

### 3.6 State Restoration

```
./smac --scenario-file <file> --restore-scenario <dir>
```

SMAC将读取指定目录中的文件，并将其状态还原为在指定的迭代处运行的保存SMAC。提供了剩下的选项（例如`--SEED`，`--OVERALL OBJ`），其设置为身体，SMAC应继续沿相同的轨迹。

此选项还可用于从SMAC V1.xx恢复运行（尽管由于MATLAB的有损性质随机调用的文件和差异，您将无法获得相同的结果轨迹）。默认情况下，状态可以恢复到迭代，这是2的权力，以及在原始SMAC运行之前的2个迭代。如果原始运行崩溃，则保存其他信息，通常允许您重播崩溃。

注意：恢复SMAC状态时，您实质上是预加载一组运行，然后运行设想。在某些情况下，如果平方例在此期间改变，这可能导致未定义的代表主义者。改变像`--Tunertime-limit`通常是一个安全的下注，但是改变了中心（如`--run-obj`）不会。

要检查可以从已保存的目录恢复的可用迭代，请使用：

```
./smac-possible-restores <dir>
```

### 3.7 Warm-Starting the Model

```
./smac --scenario-file <file> --warmstart <foldername>
```

使用与3.6节中的相同状态数据，您也可以选择使用以前的运行进行热身模型。而不是一`restore-scenario`选项使用`--warmstart`。SMAC将正常运行，但是在构建模型时，也将使用上述数据。请记住以下内容。

注意：如果执行模式咆哮，则此选项无效。

警告：由于在此版本中的SMAC中的状态恢复格式的设计限制，您不能/不应在用于加热模型的实例分发之间存在任何差异，以及我们配置的实例分发。在最佳情况下，您将在某些点（也许是`nullpointerexception`），并且在最坏的情况下，它只需用垃圾加载模型。

提示：包含的状态合并实用程序允许您轻松地合并一堆不同的SMAC进入您可以使用热情的一个状态。

### 3.8 Named Rungroups

```
./smac --scenario-file <file> --rungroup <foldername>
```

所有输出都写入文件夹<foldername>;在-Seed中运行的运行将在该文件夹中产生不同的输出文件。

### 3.9 More Options

默认情况下，SMAC只显示基本使用选项，其他选项是中间，高级和开发人员。被警告说有一堆选择，其中一些更高级和开发者选项可能会导致SMAC表现非常糟糕。

```
./smac --help-level INTERMEDIATE
```

### 3.10 Shared Model Mode (Experimental)

注意：请在决定使用此选项之前阅读此全部部分

SMAC具有一个实验选择，基本上允许多次运行SMAC来共享数据和快速构建更好的模型。

```
./smac --scenario-file <file> --shared-model-mode true
```

使用此选项运行时有几件事要记住：

首先是不同的SMAC运行需要使用必须具有的完全相同的场景

不同的种子。即使是小的无关紧要差异也可能导致这种失败（例如，如果不同机器上的位置和执行目标算法的路径是不同的）。SMAC应该在大多数情况下优雅恢复，只需忽略不兼容的运行数据，但SMAC运行可能会损坏。

2.群集之间的共享文件系统需要允许在一台机器上写入的文件

阅读另一台机器。我们有报告，在某些文件系统（AFS）上具有一些锁定策略，您无法执行此操作。在这种情况下，您仍然可以从此模式中受益，但可能需要有点粗糙。在进行第一组运行后，您可以使用-`share`运行数据执行第二批次，它们将无法读取第二批次数据，但它们应该能够读取第一批次。

3.通过 - 分享模型模式 - 频率来控制重新读取运行的频率

默认为5分钟，根据您的场景和您所需的数据量，您可能希望将其设置为更低。如果您的运行需要大于5分钟，则不需要增加这一点，只能为每一个本地算法运行重新读取数据，以防止以防止频繁地击中文件系统

4.使用SMAC内存，您可能需要增加您给SMAC的内存量

环境变量。在Bash Shell脚本中，这可以通过导出SMAC存储器= 2048和Windows Set SMAC存储器= 2048完成。

5.此模式将n个独立的SMAC运行成N依赖的SMAC。此模式设计为

帮助获得更好的执行配置，对实验目的同一分配的独立样本来处理这些运行是不合适的。可能是适当的是比较实验协议，选择在训练集上的最佳性交SMAC运行，而不是在训练集上的最佳依赖SMAC运行，并在测试集上报告它们的值。

6.此模式并不要求完全同时执行不同的SMAC。在另一个

极端情况运行可能是顺序发生的，此模式只是运行SMAC的更容易的方式，刷新运行数据，但是温暖启动模型与上一个。在另一个极端情况下，我们对一些初步实验进行了一些初步实验，所有运行都在同一时间开始。在这种情况下，中位性表现（但是看到前一点为什么这是误导性的提升，而且选择随着时间的推移选择运行，通常导致性能良好或更好。另一种可能性是有一些运行不使用此模式，并使用此模式具有此功能。根据场景，这可能允许模型保持更多多样性。不幸的是，此选项的好处和最佳实践目前是未开发的。

这种方法的一个优势是，如果您只关心快速获得良好的配置（令人担忧的重现性），您可以独立安排对集群的SMAC运行，这应该使它们更快地发货，但仍然受益于共享数据。

7. SMAC在共享模型模式下运行时，您可能会看到关于损坏文件的零星错误，

等等这些通常是安全的，无法忽略并且可能是由写作和读取同时发生的。在读取错误后，SMAC将继续尝试读取该文件。在成功读取文件之前，将不会出现进一步的错误或警告。

8.在运行结束时，您将看到一条线：

```
[info] shutdown: ./smac-output/branin-scenario/live-rundata-3.json有15个运行在关机时添加了[info]: 我们检索到至少20运行并将它们添加到
```

我们当前的数据集[Live-RunData-1.JSON => 11, Live-Rundata-2.Json => 17]

此输出表示此次SMAC（带种子3）完成15个在本地运行。它还读取11次从Run 1的运行，17次从Run 2运行。



### 3.11 Offline Validation

SMAC包括在配置过程中选择的现任者的离线评估的工具。默认情况下，给定具有N实例的测试实例文件，SMAC执行 $\approx 1000$ 个目标算法验证每个配置运行（舍入到最接近的n）。

默认情况下，SMAC限制验证中使用的种子的数量为每隔1 000种种子。这可以如以下示例中更改编号：

```
./smac --scenario-file <file> --num-seeds-per-test-instance 50
```

（此参数在实例/种子文件的情况下没有任何效果。）

#### 3.11.1 限制验证运行中使用的实例数

只使用指定的某些实例或实例种子，您可以使用`--num-test-实例`参数限制它们。指定此参数时，SMAC只能使用文件顶部的指定行数，并将继续重复它们，直到使用足够的种子：

```
./smac --scenario-file <file> --num-test-instances 10
```

例如包含种子的文件，此选项只会使用文件中指定的实例种子。

#### 3.11.2 Disabling Validation

可以跳过验证，如下所示：

```
./smac -s 拉 - 文件<文件> --seed 1 - 过渡假
```

#### 3.11.3 Standalone Validation

SMAC还包括一种验证SMAC运行之外的配置的方法。您可以使用`-configuration`选项提供配置。所有方案选项都适用于独立验证器，但检查使用屏幕以查看可用的所有选项注意：某些选项在此时不适用于验证，但无论如何都会呈现。

这是一个示例调用：

```
./smac-validate --scenario-file <file> --num-validation-runs 100  
--configuration <config string> --cli-cores 8 --seed 1
```

离线验证工具的使用说明：

- 1.此仅对测试集进行验证;不使用训练实例集。
- 2.默认情况下，此输出到当前目录;您可以使用选项更改输出目录  
**--rungroup.**
- 3.您还可以针对由`--trajectory-file`选项发出的轨迹文件验证。

## 4 File Format Reference

### 4.1 Option Files

选项文件是保存常用SMAC经常使用的不同值集的方式，而无需在每次执行中指定它们。选项文件的常规格式是配置选项的名称（没有两个破折号），一个等号，然后值（对于booleans，它应该是真或假的，小写）。目前不支持采用多个参数的选项。此外，您不能使用单个虚线的别名（例如，覆盖实验目录，您必须使用 - DIR和NOT -E）

使用选项文件时，重要的是没有两个文件（包括方案文件），指定相同的文件选项，结果配置未定义，通常这不会抛出错误。

#### 4.1.1 Scenario File

场景选项文件或方案文件是配置SMAC 3的推荐方式。SMAC中使用的场景文件与参数兼容，此处的选项名称反映该4。注意：目前不支持截止长度。

Algo围绕与输入/输出符合算法的算法算法可执行或包装脚本  
第5.1节中指定的格式。这里的字符串应通过系统shell可调用。

**execdir** Directory to execute <algo> from: (*i.e.* “cd <execdir>; <algo>”)

确定算法是否应视为确定性的算法的布尔值。

对于与发行物的后退兼容性，此选项还支持使用0 for false，而1为true。对于任何给定的实例，种子和配置，SMAC将永远不会再调用目标算法。如果设置为true，则对于任何给定的实例和配置，SMAC将永远不会再调用目标算法。

RUN OBJ确定如何将生成的输出行（如第5.1.2节中定义）转换为标量

量化如何“良好”的单个算法执行，（例如，执行执行的时间，发现解决方案有多好.....）。SMAC将尝试最小化这一目标。

目前实施的目标是以下内容：

Name	Description
<b>RUNTIME</b>	<b>Minimize the reported runtime of the algorithm.</b>
<b>QUALITY</b>	<b>Minimize the reported quality of the algorithm.</b>

RUN OBJ的总体OBJ定义了单个算法运行的目标函数，总体OBJ定义

如何组合那些单一目标以达到单个标量值以比较两个参数配置。实现的实施例如下：

3个通常会阻止您在这些文件中指定非方案选项，但通常您应该限制您的文件 to these.

这里列出的4eperry选项名称实际上是第10.5节中列出的现有选项的别名，并且完全可以使用SMAC而不使用场景文件。

Name	Description
MEAN	The mean of the values
MEAN1000	Unsuccessful runs are counted as $1000 \times \text{target\_run\_cputime\_limit}$
MEAN10	Unsuccessful runs are counted as $10 \times \text{target\_run\_cputime\_limit}$

目标运行Cuptime限制CPU时间之后，将终止单个算法执行

不成功（并视为超时）。这是一个重要参数：如果选择过高，很多时间将浪费不成功的运行。如果选择过低，优化仅偏置，仅在简单的实例上执行良好。

cuptime限制了配置允许的CPU时间的限制（即，所有算法运行时的总和，并且默认为SMAC本身的CPU时间的总和）。

壁克洛克限制了配置允许的壁锁（或实际）时间的限制。

paramfile指定具有算法参数的文件。该文件的格式在部分中介绍

#### 4.4.

outdiR指定目录SMAC应该将其结果写入。

实例文件指定包含SMAC的问题实例列表（和可能种子）的文件

在自动配置阶段。Paramiks参数实例种子文件别名此一个和格式是自动检测的。这些文件的格式在4.2节中介绍。

测试实例文件指定包含SMAC的问题实例列表（和可能种子）的文件

在验证阶段使用。Paramiks参数测试实例种子文件别名此一个和格式是自动检测的。这些文件的格式在4.2节中介绍。

功能文件指定具有实例文件中实例的功能的文件，并且可能是

测试实例文件5.第4.3节中介绍了此文件的格式。

## 4.2 Instance File Format

实例文件和测试实例文件选项使用的文件有四种潜在格式，所有这些都是CSV的第6个。在指定格式之前，重要的是要注意使用实例7指定的三种信息。

实例名称所选实例的名称。这应该对目标算法有意义

we are configuring .

实例特定信息是将传递给的免费表单文本字符串（没有空格或行中断）

执行时目标算法。

种子在执行目标算法时使用特定种子。

---

5如果存在，验证器将将功能加载到内存中以进行测试实例。6，每个单元格应该是双引用的（即“），并使用逗号作为小区分隔符。SMAC也支持旧方法

读取使用空格作为小区分隔符的文件，也不会括起值。但是，这些文件无法处理包含空格的实例名称。

在单独的文件中指定了SMAC但不是参数所需的7Features，请参见第4.3节。8generally实例名称在磁盘上引用特定文件。

可能的格式如下，并取决于您要指定的信息。

1. 每行仅指定唯一的实例名称。任何实例都没有使用，和种子将自动生成。
2. 每行指定一个种子后跟实例名称。每一行都必须是唯一的，但每个线实例名称将在选择该实例时按顺序使用其他种子。
3. 每行指定实例名称，后跟实例特定信息。每一个例子名称必须是唯一的，将自动生成种子。
4. 每行指定一个种子，后跟实例名称，后跟实例特定信息 - 合适的。每一行都必须是唯一的，此外，对于所有实例名称的实例，实例特定信息对于所有种子值必须相同（i. you无法指定作为所使用的种子函数的不同实例特定信息）。

### 4.3 Feature File Format

功能文件指定要用于实例的功能。功能文件以CSV格式指定，每行的第一列应列出实例文件中显示的实例名称。后续列应列出指定计算连续功能的双倍值。逐惯例，值-512和-1024用于表示要素值缺失或不适用。所有实例必须具有相同数量的功能。

在文件的顶部，必须出现一个标题行，在实例名称上方出现的单元格不重要，但对于每个功能，必须指定一个唯一和非数字（即它必须包含至少一个字母）的功能名称。

#### 4.4 参数配置空间格式

PCS格式需要每行包含以下3个条款中的一个，或者只有空白/注释。

参数声明条款指定参数，域和默认值的名称。

条件参数条款指定参数处于活动/非活动状态时。

禁止参数条款指定参数设置的组合是非法的。

在整个文件中允许评论;他们从一个#开头，然后跑到一行的末尾。

#### 4.4.1 Parameter Declaration Clauses

PCS格式支持两种类型的参数：分类和数字。

#### Categorical parameters

分类参数采用一个有限一组值。指定分类参数的每一行都应该是表单：

```
<parameter_name> {<value 1>, ..., <value N>} [<default value>]
```

其中'<默认值>'必须是可能值的集合之一。

### Example 1:

```
decision-heuristic {1,2,3} [1]
```

这意味着参数'决策启发式'可以给出三种可能的值中的一个，默认分配为'1'。

### Example 2:

```
@1:loops {common,distinct,shared,no}[no]
```

在这个例子中，有些密码参数名称'@

1：循环'是完全合法的;参数名称中的唯一禁用字符是空格，逗号，报价和括号。分类参数值也是具有相同限制的字符串;特别是，对分类参数值没有限制为数字。

### Example 3:

```
DS {TinyDataStructure, FastDataStructure}[TinyDataStructure]
```

如本示例所示，参数值甚至可以是Java类名（例如，通过反射）。

### Example 4:

```
random-variable-frequency {0, 0.05, 0.1, 0.2} [0.05]
```

最后，如本示例所示，通过简单地将其域（选择合理值的子集），可以通过简单地将数值参数视为分类。

## Numerical parameters

数值参数（真实和整数）指定如下：

```
<parameter_name> [<min value>, <max value>] [<default value>] [i] [l]
```

尾随'i'和/或尾随'l'是可选的。“i”表示参数是一个整数参数，'l'表示参数域应为日志转换以进行优化（参见下面的示例3和4）。

### Example 1:

```
sp-rand-var-dec-scaling [0.3, 1.1] [1]
```

参数SP-RAND-VAR-DEC-DECING具有默认值为1的实际值，并且我们可以从（关闭）间隔[0.3,1.1]中选择值。请注意，在此间隔之外可能存在其他参数值，该参数是参数的原则（例如，您的求解器可能接受参数的任何正浮点值）。您在此处指定的是自动配置过程应该搜索的范围（即，您预计先验以包含良好值的范围）;当然，指定范围内的每个值都必须是合法的。选择最优化的范围尺寸时有一个权衡。

### Example 2:

```
mult-factor [2, 15] [5]i
```

参数Mult因子是整数值，在2到15之间的任何整数值（包含），并且在技术上也有5个默认值，还可以将此作为具有可能值的分类参数{2,3,4, 5,6,7,8,9,10,11,12,13,14,15}。但是，不排序分类参数，并且使用整数参数允许配置过程使用自然顺序关系（这是有用的，因为先验，我们预期的关闭值以产生类似的性能）。

### Example 3:

```
DLSc [0.00001, 0.1] [0.01]l
```

参数DLSC具有默认值为0.01的实值，并且我们可以从（关闭）间隔中选择值[0.00001,0.1]。尾随'L'表示此参数自然地在日志刻度上变化。如果我们要使参数离散，自然选择将是{0.00001,0.0001,0.001,0.01,0.1}。这意味着，参数值0.001和0.01之间的距离与0.01和0.1之间的距离相同（在LOG10变换后，0.001,0.01和0.1分别为-3，-2和-1）。我们在“L”标志上表达了对数级的自然变化。

### Example 4:

```
first-restart [10, 1000] [100]il
```

参数First-Restart具有默认值100的整数值，我们可以从（关闭）间隔[10,1000]中选择它的值。它也在对数标度上自然变化。例如，由于这种对数刻度，在绘制的转换之后，首先重启的均匀随机值将产生低于100的时间。

## Restrictions

数值整数参数必须具有指定为整数的下限和上限，以及默认值也必须是整数。

具有日志比例的参数的界限必须严格为正数。

### 4.4.2 Conditional Parameter Clauses

根据某些“更高级别”参数的实例化，某些“较低级别”参数可能无法处于活动状态。例如，如果未选择启发式，启发式的子参数并不重要（即，活动）。默认情况下，所有参数都被视为活动状态，而条件参数条款表示参数未激活的条件。条件参数条款的语法如下：

```
<child name> | <parent name> in {<parent val1>, ..., <parent valK>}
```

如果父参数<父名称>拍摄k指定值之一，则这可以读取为“子参数<子名称>仅为Active。”在任何条件参数子句中未列为子参数的参数始终处于活动状态。参数也可以在多个条件参数条款中列为子项，并且如果满足每个此类子句的条件，则仅处于活动状态。

### Example:

```
sort-algo {快速, 插入, 合并, 堆, stooge, bogo} [bogo]快速选择 - 方法{第一, 随机, 中位数}  
[随机]快速选择 - 方法|在{Quick}中排序 - algo
```

在此示例中，快速选择方法在设置为快速的Sort-algo参数上是条件的，并且否则将被忽略。

### 4.4.3 Forbidden Parameter Clauses

禁止参数是无效的参数值的组合（例如，某个数据结构可能与不更新数据结构的懒惰启发式不兼容，从而导致不正确的算法行为）。配置方法永远不应尝试使用禁止参数配置运行算法。forbidden参数组合的语法如下：

```
{<parameter name 1>=<value 1>, ..., <parameter name N>=<value N>}
```

### Example:

```
DSF {DataStructure1, DataStructure2, DataStructure3}[DataStructure1]  
PreProc {NoPreProc, SimplePreproc, ComplexPreproc}[ComplexPreproc]  
{DSF=DataStructure2, PreProc=ComplexPreproc}  
{DSF=DataStructure2, PreProc=SimplePreproc}  
{DSF=DataStructure3, PreProc=ComplexPreproc}
```

在该示例中，存在不同的数据结构和不同的简化。DataStructure2是 - 与ComplexPreproc兼容，DataStructure2与SimplePreProc和ComplexPreproc兼容。请注意，默认参数设置不允许包含参数值的禁止的分数。

## 5 Wrappers

### 5.1 Algorithm executable / wrapper

由Algo参数指定的目标算法必须遵守以下一般合同。在修改自己的代码直接实现这方面，这是一个选项，第5.3节中概述了其他方法。

#### 5.1.1 Invocation

算法必须通过使用参数的以下命令通过系统命令行来调用：

```
<algo_executable> <instance_name> <instance_specific_information> <cutoff_time>  
<cutoff_length> <seed> <param> <param> <param>...
```

ALGO可执行文件确切地在方案文件中的ALGO参数中指定的内容。

实例名称我们正在执行的问题实例的名称。

实例特定信息与此实例相关联的任意字符串，如在其中所指定的  
姿态文件。如果没有信息存在，则始终在此处传递“0”。

截止时间允许运行目标算法的秒数为单位。这是责任

在被遵守的情况下确保这一点。没有必要的是，实际算法执行时间（挂钟时间）低于该值（例如，算法需要清理，否则才能终止某些阶段的算法）。

截止长度域特定测量算法应考虑自己完成。

种子算法应该用于种子本身的正整数（用于再现性）。“-1”在使用时使用  
该算法是确定性的

param为算法参数中指定的所选配置设置活动参数

文件。SMAC只会通过活动的参数。另外，SMAC不保证以任何特定顺序传递参数。每个参数的确切格式是：-Name value<sup>9</sup>

上面的所有参数始终通过，即使它们不适用，在这种情况下是一个假人  
值将通过。

## Environment Variables

最近版本的SMAC还设置了以下环境变量，不应将其视为求解器输入，但以某种方式涉及执行。除非您需要一些高级功能，否则可以完全忽略此部分时，您可以完全忽略此部分。

---

<sup>9</sup>目标算法将该值视为单个参数，即使它包含空格，也将被视为更多  
而不是一个参数，即在shell中执行这个问题，你会看到-name'值'，以确保将该值传递为单个参数。旧版本也通过了单句话



Environment Variable	Purpose
<b>AEATK_CONCURRENT_TASK_ID</b>	A zero indexed value of which concurrent run SMAC is executing. No two concurrent runs will see the same value, but subsequent runs will see the same value. This is mainly intended to allow the wrapper to manage CPU affinities.
<b>AEATK_SET_TASK_AFFINITY</b>	This environment variable is NOT set by SMAC, or used by SMAC but is used internally by some wrappers to ensure that AEATK_CONCURRENT_TASK_ID is read and the task is tied to a specific core. It is strongly recommended that you set this value to “1” which your wrapper then reads to know to set the affinity properly. This isn’t enabled by default, because some clusters, notably SGE do not set job affinities properly and so parallel jobs will get tied to the same core.
<b>AEATK_PORT</b>	The wrapper can send in progress up dates to SMAC to the localhost via UDP on this port. The message format is a single double value. While SMAC doesn’t directly use this presently, other utilities such as algo-test do, and future versions may, and some advanced options may preform better with this set.
<b>AEATK_CPU_TIME_FREQUENCY</b>	Signifies how often updates to the runtime should be sent. This is only a hint, and roughly should be treated as: there is no point in sending updates more frequently than this value in seconds.
<b>AEATK_EXECUTION_UUID</b>	A UUID associated with the particular invocation of the wrapper. The primary purpose of this is facilitate identify every process associated with a specific invocation of a wrapper on a particular computer. This is primarily used in conjunction with the <b>-cli-kill-by-environment-cmd</b> option. If this environment variable already exists, then another one will be chosen, so do not rely on this particular variable being set. You should ensure that environment variables are passed to all sub processes however, so that they can be killed accordingly.

### 5.1.2 Output

目标算法可以自由地输出任何内容，这将被忽略，但必须在某些点输出以下格式10中的行（仅一次）：

此算法运行的结果：<status>, <运行时>, <runlength>, <quality>,  
<seed>, <additional rundata>

状态必须是SAT（表示成功的运行），unsat（表示成功

运行这是不可挑离的），超时如果算法在分配的时间内没有完成，则如果在算法运行期间发生的事情发生了不足，或者如果某些东西阻止目标算法成功执行，则崩溃，并且据信进一步的尝试将是徒劳的。

SMAC不会区分SAT和unsat响应，并且这些初级使用是历史，并且作为算法通过输出是否正确执行该算法正在执行

---

还允许有10 other字符串，但有一天会被替换。最值得注意的是“结果是发行者：”

有问题的实例是满意的。有关如何利用此功能的信息，请参阅`--Verify-sat`选项。注意：如果包装器报告SAT和ONSAT跨运行的相同实例，则默认情况下默认崩溃。偶尔边缘的突出的参数速度跳闸并转动求解器越野框，因此存在这种安全防护装置，以帮助检测是否发生这种情况。要更改此行为，请使用`--check-sat-consighary`和`-check-sat-suredency-exception`选项。

SMAC还支持报告SAT和不可履行的令人满意和成功。注意：这些只是别名，SMAC不会保留在日志或状态文件中哪个别名。

中止在目标算法找不到所需文件的情况下可以是有用的，或者权限问题可防止对其访问。这将导致SMAC立即停止运转。使用此选项使用此选项，只有在算法知道后续结果可能失败时才会报告。对于像零星的网络故障等宇宙射线引起的故障等的东西，应该考虑使用与`--retry-retry-clashed-count`和`-abort-in-clash`选项的崩溃，以减轻这些。

在其他文件或日志中，您可能会在使用以下附加类型后看到以下其他类型。运行，它表示当前正在运行中间的结果，并杀死它表示在内部决定在完成之前终止运行的SMAC。这些仅为内部值，不允许包装器输出这些值。如果包装器报告这些值，则将被视为运行的状态崩溃。

运行时运行此算法期间使用的CPU时间的量。SMAC不测量CPU时间

直接，这是与TUNERTIMEOUT的使用量。当报告11时，您可能会出现意想不到的性能下降。

注意：运行时应始终严格少于报告Sator Unsat时的请求的截止时间。运行时必须严格大于零（而不是NaN）。

如果算法报告超时或崩溃，则算法可以报告所使用的实际CPU时间，并且SMAC将正确地将其视为优化目的的超时，但是计算实际时间`--Tuntrime-Limit`目的。

RunLength算法进展多远的域特定度量。此值必须来自集合：

$$-1 \cup [0, +\infty).$$

质量域特定测量质量的质量。此值需要来自集合：

$$(\quad, +\quad).$$

注意：请记住，SMAC将尝试最小化此值。如果您希望最大化此值，则您的包装器应从已知的某些常数减去您的数量，以大于任何值。

注意：在某些情况下，例如在模型中使用日志变换时，此值必须是： $(0, +\infty)$ 。

种子在该目标算法执行中使用的种子值。

---

11通常在定位非常短的算法时发生的，该算法具有未计算的大开销。

注意：根据2.06.02版，SMAC忽略此种子参数。SMAC将始终在内部使用所请求的种子，因此您可以忽略此输出。注意以前的版本，以及Paramills和其他应用程序仍可能要求这匹配。

附加rundata将与之关联的字符串（不包含逗号或换行符）

就SMAC而努力。此字符串将保存在运行和结果文件中（第6.2节）。注意：额外的rundata在撰写本文时与Paramills不兼容，因此如果您希望兼容，则包装器不应包含此功能或前面的逗号。

除其他rundata之外的所有字段都是强制性的。如果该字段不适用于您的场景，则可以替换0。

## 5.2 Wrapper Output Semantics

由于SMAC完全由包装器执行的目标算法绝缘，因此由包装器达到包装纸，以确保强制执行关于截止值和运行长度的约束。偶尔包装器可能无法正确强制执行这些约束，SMAC需要以某种方式处理这些情况。下表概述了SMAC如何将值转换和详细信息SMAC各个部分中使用的值。在未来版本中，此表的某些部分可能会发生变化，因此最好确保您的包装器表现良好。

注意：表中的截止时间是SMAC计划的时间量，方案截止时间表示为  $\kappa_{max}$ 。

status	cutoff ( $\kappa$ )	runtime ( $r$ )	Tuner Time	PAR10 Score	Model
*	*	$(-\infty, 0)$	EXCEPTION THROWN		
ABORT	*	$[0, \infty)$	EXCEPTION THROWN		
CRASHED	*	$[0, \infty)$	$r$	$10 \cdot \kappa_{max}$	$10 \cdot \kappa_{max}$
SAT, UNSAT	$\kappa \leq \kappa_{max}$	$[0, 0.1]$	0.1	$r$	$r$
SAT, UNSAT	$\kappa \leq \kappa_{max}$	$[0.1, \kappa)$	$r$	$r$	$r$
SAT, UNSAT	$\kappa \leq \kappa_{max}$	$[\kappa, \kappa_{max})$	$r$	$r$	$r$
SAT, UNSAT	$\kappa \leq \kappa_{max}$	$[\kappa_{max}, \infty)$	$r$	$10 \cdot \kappa_{max}$	$10 \cdot \kappa_{max}$
TIMEOUT	$\kappa < \kappa_{max}$	$[0, \infty)$	$r$	$\kappa$	$\kappa$
TIMEOUT	$\kappa = \kappa_{max}$	$[0, \infty)$	$r$	$10 \cdot \kappa_{max}$	$10 \cdot \kappa_{max}$

位置的描述如下：

调谐器时间将从剩余调谐器时间限制中减去的时间量

scenario.

PAR10评分将用于配置之间的经验比较的值。

模型将用于构建模型的值。

## 5.3 Wrappers & Native Libraries

为了优化算法，SMAC需要一种调用它的方法。在修改代码以管理时序和手动输入机制的同时，这有时可以是侵入性的并且难以管理。有三种其他方法可以考虑使用。

包装符可动性脚本，可自动管理资源并格式化指定的字符串

进入可通过实际目标算法可用的东西。这种方法可能是最常见的，但它的缺点是它们经常依赖于第三方脚本语言，并且对于较小的执行时间具有大量的开销，可能不会因TunerTimeOut限制而被解释。SMAC中包含的大多数示例都使用此方法，包括的包装器可以适用于您自己的项目。

注意：在编写包装器时，重要的是不要轮询目标算法的输出流，特别是如果有很多输出。这样做经常导致锁争用，并显著修改算法的运行性能足以使结果配置不适当地调整到真实算法的性能。

在SMAC V2.06.02中引入的INTER过程通信，可以通过--TAE选择IPC TAE

选项。它基本上使用各种形式的进程通信来通知跑步的其他进程，并等待它响应。目前，唯一的现有机制超过UDP，但其他方法是可能的。例如：

```
./smac --scenario-file <file> --seed 1 --tae IPC
--ipc-mechanism UDP --ipc-remote-port 5050
--ipc-remote-host localhost
```

将使用将所需信息发送到端口5050上的localhost。然后它将等待响应，并将该响应解析为字符串。有关详细信息，请参阅附录中的“进程间通信目标算法评估器选项”部分。

目标算法评估员这可能是最强大的，也是最复杂的方法。

SMAC以一种方式归档，使其相当简单地替换与完全自定义的东西执行的机制。这可以通过创建有型级视网膜图界面的新实现，而无需重新重新编译SMAC，这负责将RUN请求（Runconfig对象）转换为运行结果（algorithm Run对象）。输入和输出对象都是简单的值对象，因此SMAC之间的耦合和代码的其余部分几乎为零。有关更多信息，请参阅??

## 6 Interpreting SMAC's Output

SMAC将各种信息输出到日志文件，轨迹文件和状态文件。大多数文件都是人类可读的，本节介绍这些文件。注意：所有输出都将在-Rungroup子目录中写入outorir。

### 6.1 Logging Output

SMAC使用SLF4J (<http://www.slf4j.org/>)，该库允许轻松抽象和更换日志系统，并使用LOGBACK ([http://logback.qos.ch](http://logback.qos.ch/))作为默认日志记录系统。虽然有限的能力通过命令行更改日志记录选项（例如 - -log级别， - --console-log-lock级别， - -log-all-call-strings， - -log-all-process-output）。通过设置系统属性<sup>12</sup>，可以使用-dlogback.configurationFile = / path / to / config.xml来覆盖配置

---

<sup>12</sup>您必须编辑启动脚本smac或smac.bat

注意：如果您在SMAC中替换记录器或修改配置文件，则日志记录命令行选项可能不再有效。

默认情况下，SMAC将以下日志文件写入磁盘（注意：n表示已复制的设置）:

log-run.txt一个日志文件，该文件包含记录所有信息的完整转储，以及记录的位置  
from.

log-warn.txt包含与上述文件相同的信息，除非仅来自警告和更高级别  
messages.

log-err.txt包含与上述文件相同的信息，除非仅从错误消息。

### 6.1.1解释日志文件

SMAC在执行时基本上通过三个阶段：

读取设置阶段输入文件，验证其参数。执行的一切必要  
构造自动配置阶段。此阶段结束，一旦出现以下消息：

SMAC开始于：2014年4月10日10:01:40。尽量减少惩罚的平均运行时（PAR10）

自动配置阶段：SMAC现在正在积极配置目标算法。SMAC将  
花费大部分时间在这里，并输出它的进步。

您将在此处看到两种类型的消息：

1) 现任事件改为：配置2（内部ID：0x7），惩罚平均运行时（PAR10）：7.1；  
基于2次运行估计。样本呼叫新的现役配置2（内部ID：0x7）：

```
cd saps; ruby saps_wrapper.rb instances/train/SWlin2006.19724.cnf 0 10.0 2147483647 -1  
-alpha '1.1' -ps '0.1' -rho '0.84' -wp '0.06'
```

这表示现任者（到目前为止发现的最佳配置）已更改为配置2（此ID用于SMAC输出的某些文件）。它还给出了  
在我们已经看到的实例上的这种配置性能的样本估计。

接下来，给出了这种配置的示例呼叫，以便您可以自己测试。根据此，您可以确定所选的实际配置，在此示例  
中是：

*-alpha '1.1' -ps '0.1' -rho '0.84' -wp '0.06'*

2) 更新的估计抵制平均运行时（PAR10）相同的现任：3.86;现在基于4次运行估计。

由于SMAC继续运行，它将通过制作更多的现任者配置样本来继续炼制估计，偶尔会为您提供更新。当SMAC更  
多地了解您的实例分发时，此数字可能会变化。

一旦SMAC完成，它将输出一些摘要统计信息：

```
=====
=====
smac完成。原因：已达到总CPU时间限制（600.0秒）。SMAC的最终现任名：配置45（内部ID：0xBB），
估计罚款平均运行时（PAR10）：0.12，基于5次培训实例的5次运行。

执行的总运行总数：171，使用总计CPU时间：604秒，
使用的总壁锁定时间：607 S，总配置尝试：113。
=====
```

第一行表示为什么SMAC终止，在这种情况下，超出了CPU时间限制。它提供了对象的更新估计，在这种情况下为0.12。

离线验证阶段，取决于所用的选项，这也可能需要大量的SMAC运行。这里的逻辑实际上非常简单，因为它在很大程度上只需要运行许多算法运行和计算它们的目标。

```
-----
-----最小化惩罚平均运行时（PAR10）：最终时间：607配置45（内部ID：0xBB）：0
.14在测试集上

最终现任者的样本呼叫：CD SAP;Ruby SAPS_WRAPPER.RB实例/火车/SWLIN2006.19724.CNF 0 10.0 2147483647 -1

-alpha '1.1' -ps '0.1' -rho '0.84' -wp '0.06'

有关运行的其他信息：SMAC-Output / Run1 /
-----
```

## 6.2 State Files

状态文件允许您在执行情况的特定点检查和潜在地恢复SMAC状态。文件将写入状态 - runn / sub-directory，其中n是-seed选项的值。

所有文件都将以下惯例作为后缀或崩溃后跟迭代数字M，或在某些情况下快速或快速的bak。  
每个迭代M保存状态，其中 $m = 2n, n \in \mathbb{N}$ ，此外，smac时它会保存完成是否成功或由于崩溃。  
以下文件保存在此状态目录中（忽略后缀）：

Java obj dump-v2-itm.obj存储（Java）现任者和随机对象的序列化版本状态。一般来说，无需查看此文件，并且它不是人类可读。

paramstrings-itm.txt将每个配置的人类可读设置存储在一起，具有前缀配置的数字ID（在日志中使用以及其他状态文件）。

UNIQ Configurations-M.CSV存储配置ran更简洁但有效地联合国可读形式。第一列再次是配置的数字ID（在日志中使用以及其他状态文件）。

运行和结果-ITM.csv存储SMAC完成的目标算法的每次运行结果。这前13列（标题行被设计为向后兼容SMAC版本1.xx.每列都标有其包含的数据，下列值应使用一些描述。

实例ID这是使用实例，是实例文件选项中指定的第n个实例名称。

响应值 (y) 这是运行OBJ上的值确定的值。

审查指示截止时间是否低于原始的截止时间  
跑步。0表示假，1表示真实。

运行结果代码这是从运行结果到整数的映射，以与以前的版本一起使用。

param-file如果启用了-save-context，则paramfile的副本将位于状态文件夹中

实例如果启用了--Save-上下文，实例文件的副本将在状态文件夹中

实例 - 如果启用-save-上下文，并且SMAC运行功能，则副本  
功能文件将位于状态文件夹中。

如果启用--Save-上下文，并且SMAC正在使用方案文件，则是 - 索取文件的副本  
将在状态文件夹中。

### 6.3 Trajectory File

SMAC还将轨迹文件输出到详细的traj-run-n.csv中，并在执行过程中概述现任（按ID），并概述其性能。第一行给出--Rungroup，然后是 - 它们。

其余文件遵循以下格式：

Column Name	Description
CPU Time Used	Sum of all execution times and CPU time of SMAC
Estimated Training Performance	Performance of the Incumbent under the given <b>-run-obj</b> and <b>-overall-obj</b>
Wallclock Time	Time of entry with respect to wallclock time.
Incumbent ID	The ID of the incumbent as listed in the <b>param.strings</b> file in §6.2, and the logs
Automatic Configurator (CPU) Time	CPU Time used of SMAC
Full Configuration	The full configuration of this incumbent

注意：SMAC还输出TRAJ-RUN-N.TXT前五列是相同的，但剩下的列表示配置，每个单元格是一个键和值。这是相同的，该轨迹文件由参议员输出。

### 6.4 Validation Output

完成验证完成四个文件时，（再次n是--sey参数的值）。最后根据使用哪些选项，尤其是SMAC验证，输出的实际文件名可能会有所不同。

1. ValidationResults-traj-run-n-waltime.csv: 包含摘要的CSV文件  
验证结果，验证配置ID映射到下一个文件中的一行
2. validationCallStrings-traj-run-N-walltime.csv: A CSV file containing a map-  
验证配置ID之间的PING到实际配置，以及示例呼叫字符串。

3. `validationPerformanceDebug-traj-run-N-walltime.csv`: A CSV file that contains 详细分解了最终验证得分如何获得。此文件适用于人类消费，而不是解析。
4. `validationObjectiveMatrix-traj-run-N-walltime.csv`: A CSV file that contains 表格，每个行（配置）问题实例种子对的目标，如列中给出的。
5. `ValidationRunmatrix-traj-run-n-waltime.csv`: 包含表的CSV文件，对于每行（配置）从包装器（忽略前缀）的响应。

## 6.5 JSON Output

对于每次运行SMAC，SMAC还将输出文件`Live-Rundata-n.json`。文件的格式包括所有问题实例的数组表示。然后，另一个运行是一个接一个地报告。实际的JSON表示表示对象的主要语义视图，足以构建所有运行数据。

# 7 Developer Reference

本节旨在作为需要修改SMAC代码基础的人的指南。

## 7.1 Design Overview

SMAC应用程序分为三个不同的项目，如下所示：

SMAC包含特定于SMAC的所有逻辑（例如，SMAC算法，施工SMAC对象）。实质上，它缝合了自动配置器库的组件。这些来源包括在`SMAC-SRC.JAR`中。

算法执行和抽象工具包包含所有主要的抽象/模型

SMAC（例如，执行算法的实例，目标算法配置和方法，.....）。SMAC在这个库中使用寿命的90%的代码。它还包含用于将这些抽象中的数据转换为构建模型所需的输入的代码。这些在`ACLIB-SRC.JAR`文件中用SMAC发货。注意：历史上，这被称为“ACLIB”，在此版本的SMAC内部，代码仍称为ACLIB。下一个版本的SMAC可能会重命名它。

随机森林随机森林模型代码。这些来源包含在`Fastrf-src.jar`中。

本文档的范围仅管理前两个项目。在撰写算法时执行和抽象工具包没有发布的文档，但如果您通过电子邮件发送上述版本可用。

在四个课程中运行SMAC所需的大部分代码

`SequentialModelBasedAlgorithmConfiguration`, `SMACBuilder` and finally, `SMACExecutor`.



## 7.2 Class Overview

SMAC最重要的课程如下：

Name	Description
AbstractOptions	Base class for creating new options for SMAC. While not important in and of itself, you will generally be implementing or modifying one of it's subtypes to implement options.
AlgorithmRun	Interface that represents the results of a target algorithm run. These are created by a TargetAlgorithmEvaluator. Outside of TargetAlgorithmEvaluator 这些类通常是可变的。
AlgorithmExecutionConfig	Immutable object containing the information required to invoke a target algorithm.
InstanceSeedGenerator	Interface that gets seeds for a ProblemInstance. These objects are constructed by ProblemInstanceHelper
ModelBuilder	Interface whose implementations should result in a constructed model.
OverallObjective	Aggregates many RunObjective values under some statistic (e.g.mean), to produce a value to be optimized.
ParamConfiguration	Class that represents a specific setting of the target algorithm's parameters. This class also implements the Map interface, though does not support all the required operations. The ID associated with is object, is used only for logging and should not be used in the code. Finally although this class is not immutable the general life cycle is that the object is created, given specific values, and then never changed again. In future this may be augmented with the ability to prevent writes. These objects are always constructed via the ParamConfigurationSpace.
ParamConfigurationSpace	(Almost immutable) class that represents the entire configuration space of a target algorithm. This class is constructed with the <b>Algorithm Parameter File</b> described in section 4.4. This class also contains the specifics of each parameter (e.g.domains, defaults, etc...). Currently the Random object used is the only portion that is mutable, and this will change in the future.
ProblemInstance	Immutable class that represents a specific problem instance, constructed by ProblemInstanceHelper.
ProblemInstanceSeedPair	Immutable class that represents a problem instance and seed. Decisions of which seed to use when scheduling a run are made in RunHistory.
RunConfig	Immutable class that represents a problem instance seed pair, and configuration to execute.
RunHistory	Interface that saves all the runs performed, and allows various queries against this information.
RunObjective	Converts an AlgorithmRun into a scalar value for optimization
SanitizedModelData	Converts the run data into a format to use when building the model. Other things such as PCA, and other data filtering are done here. This interface and mechanism will likely be refactored in the future as it is brittle at the moment.
SeedableRandomSingleton	A global random object whose existence is a convincing case that Singleton's are Anti-Patterns. This will, thankfully, go the way of the dodo bird at some point.
StateFactory	Interface that constructs StateSerializer & Statedeserializer 分别管理节省和恢复状态。
TargetAlgorithmEvaluator	Interface whose implementations should be able to run the algorithm (i.e. Implementations should convert RunConfig objects to AlgorithmRun objects). See section ?? for more information.

SMAC Library Classes	
Name	Description
AbstractAlgorithmFramework	<i>Non-abstract</i> class that provides a default Automatic Configurator (ROAR)
SequentialModelBasedAlgorithmConfiguration	Class that subtypes AbstractAlgorithmFramework并实现SMAC所需的方法
SMACExecutor	Parses command line options and creates some of the objects SMAC needs to execute (SMAC entrypoint)
SMACBuilder	Takes the options parsed by SMACExecutor or some other utility, and builds everything necessary to create an instance of AbstractAlgorithmFramework. If you want to plug smac into your application, you generally want to mimic what SMACExecutor does to invoke SMACBuilder.
Validator	Performs Validation of selected configurations
ValidatorExecutor	Entry point to stand alone validation utility

### 7.3 算法执行和抽象工具包

在一些未来的一点上，将在临时提供更好的指南，请通过电子邮件向上述作者发送电子邮件。

## 7.4 Running SMAC in Eclipse

根据您希望的内容，可能更好地要求Git存储库访问，其中包含Ant构建脚本和现有的Eclipse项目。但以下过程将为Eclipse提供工作安装。

注意：本指南相当直接，除了步骤10和11，所以如果您舒适Eclipse您应该只是跳到这些步骤。

要在Eclipse中启动SMAC项目项目执行以下操作：

1. 在Eclipse中创建一个新项目，确保JDK为1.7或更高。
2. Create a new source folder: aeatk
3. Create a new source folder: smac
4. Create a new source folder: fastrf
5. Create a new folder: lib
6. 将SMAC的Lib文件夹中的所有JAR文件复制到Eclipse的lib文件夹中  
Project, 除SMAC.JAR, SMAC-SRC.JAR, AEATK.JAR, AEATK-SRC.JAR, FASTRF.JAR和FASTRF-SRC.JAR。
7. 将smac-src.jar解压缩到SMAC源文件夹中

8.将Aeatk-src.jar解压缩到Aeatk源文件夹中

9.将fastrf-src.jar解压缩到FAStrf源文件夹中。

10.右键单击项目并转到“属性”→Java编译器→注释处理和支票  
启用项目特定设置并启用注释处理。

11.然后在项目属性中导航到Java Compiler→注释处理→出厂路径  
然后点击添加jar然后选择lib / spi-0.2.4.jar并点击确定。

可以从shell脚本文件夹中检索任何应用程序的入口点，例如通过打开smac文件，我们可以看到输入点是：ca.ubc.cs.beta.smac.executors.smacexecutor和smac-validate是：ca.ubc.cs.beta.smac.executors.validatorexecutor。

许多辅助实用程序包含在ca.ubc.cs.beta.aeatk.example中。子包。

注意：如果您尝试使用SMAC打包的现有场景，它们包含相对于root的路径  
smac dir。因此，在您的运行配置中，您应该将工作目录设置为某些SMAC Release的根目录以在命令行上运行它。

如果在运行SMAC时，您会看到以下任一错误：警告：我找不到类路径上的任何目标算法评估器。如果你自己制作了  
这个jar，你就没有正确设置SPI的机会。有关更多信息，请参阅Aeatk手册/开发人员参考，没有找到名称的目标算法评估  
板：CLI

这意味着您没有正确遵循步骤10和11。

## 8 Acknowledgements

我们感谢Jonathan Shen将我们的随机林代码从C移植到Java，以准备所有SMAC的Java港口。Alexandre Fr'echette和Chris  
Thornton的持续反馈和修补程序。我们还要感谢Marius Schneider为许多有价值的早期错误报告和改进建议。

## 9 References

- [1] Hutter, F., Hoos, H. H.和Leyton-Brown, K. (2011A)。贝叶斯响应审查的优化  
数据。2011年贝叶斯优化，顺序实验设计和匪徒的尼斯研讨会。在线发布。
- [2] Hutter, F., Hoos, H. H.和Leyton-Brown, K. (2011b)。基于常规的常规模型优化  
算法配置。在proc. Lion-5, LNC, 第507-523页。
- [3] Hutter, F., Hoos, H. H., Leyton-Brown, K.和Stutzle, T. (2009)。发行方法：自动算法  
配置框架。人工智能研究杂志, 36: 267-306。

## 10 Appendix

### 10.1 Return Codes

Value	Error Name	Description
0	<b>Success</b>	Everything completed successfully
1	<b>Parameter Error</b>	There was a problem with the input arguments or files
2	<b>Trajectory Divergence</b>	For some reason SMAC has taken a unexpected path ( <i>e.g.</i> SMAC executes a run that does not match a run in the <b>-runHashCodeFile</b> )
3	<b>Serialization Exception</b>	A problem occurred when saving or restoring state
255	<b>Other Exceptions</b>	Some other error occurred

注意：除255之外的所有错误条件都是固定的。然而，在未来的一些例外之前报告的255可以根据需要/要求改变为非255个价值

#### 10.2 java smac的版本历史记录

2.00版本2.00（2012年8月）java smac的首次内部发布（这是原始的端口和扩展 Matlab version).

版本2.02（2012年10月）第一次公开发布SMAC v2，并包含来自上一个的许多修复 release.

2.04版（2012年12月）java smac的第二个发布包括以下改进：

- 1.验证文件输出时间与调谐次数一致
- 2.已将某些信息日志语句移至调试和追踪的一些调试
- 3.添加了验证SAT和UNSAT的响应是否与实例一致的支持  
具体信息请参阅 - VERIFYSAT选项以获取更多信息
- 4.添加了对SMAC内存环境变量的支持，以控制RAM的数量（in MB）SMAC将在通过提供的shell脚本执行时使用。
- 5.现在上下文将添加到状态文件夹中，以便在稍后更轻松地调试问题，以禁用咨询 the **--saveContext** option.
- 6.大大提高了状态序列化代码中的内存使用情况，现在我们释放现有模型在建立新的之前，对于某些JVM来说，这可能会提高内存使用情况。

版本2.04.01（2013年2月）java smac的小错误修复

- 1.添加了验证培训集实例的选项

- 2.现在可以使用<默认>作为配置以验证
- 3.修复了在我们所请求的截止时间下面的超时运行的错误，而不是正确计算  
considering incumbent changes
- 4.现在可以用--initialIncumbent选项指定初始现任。
- 5.壁克服时间现在保存在轨迹文件中而不是-1

## 6. FAQ Improvements

7. Git Commit Hash现已在手动，常见问题解答和版本字符串中记录
8. (beta) 支持BASH自动完成SMAC和SMAC验证的论据。你  
可以通过运行加载文件：

```
./util/bash_autocomplete.sh
```

版本2.04.02 (2013年8月) java smac的小错误修复

- 1.关闭时现有性能显示验证时显示。
2. - 不再只是展示。

版本2.06.00 (2013年8月) 重要的功能增强功能

- 1.新的算法测试实用程序允许轻松调用包装器。
- 2.新的验证方案实用程序在方案文件上预先重建额外验证。
- 3.如果配置空间耗尽，则方案现在结束
4. SMAC现在让您只搜索一个良好的配置子空间
- 5.验证输出格式随标题的改进
- 6.始终与最初的现任者进行比较 (以防止早期的选择  
derailing the run) (See **--always-run-initial-config**)
- 7.如果运行行为现在提供不同的答案，SMAC会报告错误
- 8.新的 - restore-scenario选项使恢复方案更容易
- 9.新的--warmStart选项可以使用额外的SMAC运行预装模型。
- 10.现在可以将种子设置为SMAC的不同部分
- 11.运行时统计数据和终止原因现在重写
- 12.新的验证选项 - Validate-all, - 仅限于If-Tunertime-anted (参见验证  
所有这些选项)
13. SMAC现在在计划运行之前检查限制，而不是在运行之后立即  
之前的版本。(这意味着如果上次运行过来，但改变了它将被记录的现任。)
- 14.现在可以确定的实例可以确定 (这是按顺序排列的  
instance file via **--deterministic-instance-ordering**.)
- 15.通过新的帮助级别改进的用法，该级别与--help级和新的使用屏幕显示。
- 16.改进Bash自动完成。
- 17.目标算法评估员现在采取选项。

- 18.修复SMAC中的CPU时间计算。
- 19.清理的示例方案，提供了新的。
20. SMAC应该更加宽容方案文件中的相对路径。
- 21.现在支持的默认选项文件（SMAC将从 `/ .aclib / smac.opt`， `/ .aclib / tae.opt`读取和 `/ .aclib / help.opt`。它还可以从可用的插件读取默认值。注意：将来的版本更改为 `/ .Aeatk /`。
- 22.运行组名称现在可配置。
- 23.清理一些对象的记录。
24. Windows启动脚本，并改进Unix启动脚本。
- 25.修复了启动Unterminating子过程的包装器的锁定问题。
26. Fixed ConvergenceException error message.
- 27.现在的选项具有主要的非骆驼案例格式。
- 28.手动现在有一个基本选项部分，在列出所有选项之前。
- 29.对目标算法评估人员的重要API更改，因此需要先前的插件重构（另一个变化将在v2.06或v2.08中出现）。
30. SMAC现在将匹配CREAPER串联字符串中的单词的大写。
- 31.新的 - 过验证 - 种子选项应该在SMAC结束时导致验证表现出来与独立效用相同。

版本2.06.01（2013年10月）次要错误修复java smac的发布

- 1.修复了2.06.00中引入的错误导致验证验证实例分发而不是测试实例分发。
- 2.解决方案质量优化的默认采集功能现在预期改进 (instead of Exponential Expected Improvement).
- 3.固定异常如果方案文件没有扩展名。
- 4.新的选项 - 删除将导致SMAC在下一组之前中止过程如果指定的文件删除，则运行（好像已击中它的CPU时间限制）。
- 5.新选项 - runs-on-file-delete将导致smac杀死正在进行的任何运行。这个选项应谨慎使用，因为它可能会导致SMAC选择错误的现任，它应该始终与--terminate-op-ople。
- 6.新选项 - 遍历运行 - 每次迭代都会导致SMAC输出运行和结果文件每次运行都需要恢复状态。如果您的群集或环境特别不可靠，这非常有用。当场景中的运行时，不应使用的时间非常小，因为时间SMAC将写入磁盘，从 $O(n)$ 到 $O(n^2)$ 的变化，其中 $n$ 是它执行的运行数量。
- 7.如果SMAC正在关闭意外的原因（例如OutOfMemoryError，或者它收到了SIGTERM），SMAC现在将尽最大努力用“关机”前缀写入最终批次的状态数据。注意：此状态可能因各种原因损坏，即使它正确写入您可能无法正确恢复，因为快照拍摄可能来自迭代的中间。

---

13分配迭代次数与运行数量线性缩放。

- 8.修复错误消息错误消息，错误地报告了缺少的情况，实际上这是缺少的测试实例。

版本2.08.00（2014年8月）可用性和验证变更

1. SMAC现在更挑剔的实例名称和功能名称匹配。
- 2.新的SAT检查实用程序允许确定实例的每个实例的可靠性  
file.
- 3.在执行包装时，现在设置变量Aeatk并发任务ID -  
每个，包含索引到并发作业的数量。这主要用于允许包装器正确确定CPUIf。有关更多信息，请参阅包装部分。
4. SMAC已经令人遗憾地说明了。默认级别信息现在只包含  
最终信息，以及有关现任事件的更改的信息。Debug包含大多数旧信息级别，跟踪包含大多数旧调试级别。  
旧的跟踪水平从未使用过，已被删除。
- 5.现在可以使用--instances和-test-virces选项由文件夹指定实例。你  
可以限制通过--instance-后缀和-test-instance-suffix使用哪些实例
6. - EXEC-DIR选项现在默认为当前工作目录。
- 7.新选项--USE-use issues将使用虚拟实例而不是实例文件（有用  
black box optimization).
- 8.新的高级选项 - 分号模型模式可能会在某些情况下提高性能，请参阅  
Section 3.10.
9. [Beta]目标算法评估员实现允许与TAE使用使用  
UDP/TCP (more to come).
- 10.在审查早期运行的配置可能会成为审查的配置  
现任者错误地。它仍然是次优，但事实上它可能永远不会发生。请参阅第10.3节中的已知问题#1。
- 11.验证舍入模式现在更改了确定性运行的运行数量，或运行  
设置问题实例种子对。
- 12.新选项--CLI-kill-by-entional-cmd允许通过环境终止所有进程  
多变的。有关更多信息，请参见第5.1.1节。
- 13.目标算法不再看到引用参数值的参数。选项 - cli-call-  
可以使用params-with引用来获取旧行为，此选项将来可能会被删除。
- 14.新选项--Quick-Seves控制是否制作任何快速保存状态。
- 15.新选项 - intermediary-saves控制是否在SMAC时全部保存许多状态  
正在运行（如果False SMAC仍将在最后保存信息）
16. Revamped Quickstart guide
- 17.验证后SMAC打印正确的终止原因消息。
18. SMAC现在将在过早退出时终止所有未完成的运行（例如由于  
CTRL+C)



19. 标准化方案选项（没有新的），但方案选项可用于Paramils  
versions 2.3.7 and later.
20. 导致确定性实例的缓解错误将永远从文件加载。这可能是  
在某些情况下发生，如果要素名称和实例文件名并不完全匹配。
21. 自动检测恢复方案选项现在在缺少文件中创建了更强大的功能
22. 如果合并无需为每个实例运行的运行，状态合并实用程序不再崩溃
23. 重命名为AepOlkkit对Aeplib的许多引用，以反映工具包的名称  
SMAC is built with.
24. 现在从 / .aeatk而不是 / .aclib读取默认选项。
25. 修复了Windows上的绝对路径未正确处理的问题。
26. 验证现在默认情况下执行1个运行1，而不是1000后的下一个倍数。
27. 现在可以指定SMAC验证将使用的核心数（仅在使用本地时  
命令行），使用--validation-cores选项。
28. 之前的状态文件夹现在重命名为保留运行名称的内容，并且不再是一个  
warning.
29. 在许多地方强调SMAC最大限度地减少客观职能。
30. SMAC现在完全忽略了包装器的响应中的种子输出（它自动分）  
阐述了所要求的价值。如果您的包装器未正确设置此值，则可能会注意到SMAC中的差异。
31. 已弃用并删除了一些验证选项
32. 现在可以使用--trajectory-files选项在一个传递中验证多个轨迹文件。
33. 验证的输出格式已完全改变为更有用。
34. Traj-run-n.csv现在详细介绍了traj-run-n.csv，并且略有不同  
mat.
35. SMAC现在需要Java版本7来运行。
36. 不再使用conf / logback.xml，并且该文件在内部存储。覆盖配置 -  
汇编，设置Java System属性Logback.configurationFile = / path / to / config.xml
37. 轨迹文件中的某些列已被重命名为清晰度。订单仍然是一样的。
38. 将默认包装器字符串更改为“运行此算法的结果：”。

### 10.3 Known Issues

1. 在罕见的情况下，最初被拒绝后由SMAC重新选择的配置可能会继续  
他们否则不应该的挑战。如果配置继续取得挑战 -  
完全，在成为现任之前，我们将目前检查所有的运行，这比必要的严格更昂贵。
2. 使用-showhiddenparameters, --help, 或 - versions的任何别名作为其他参数（例如，  
setting --rungroupname --help）不会正确解析（这不太可能固定，直到有人抱怨）。

- 3.在连续积分参数中使用大参数值可能导致精度损失，或如果值太大，崩溃。
- 4.如果不是所有实例都有功能，则会发生ArrayOutOfBoundsException
5. - 纳米种子 - 每次测试 - 实例和--Num-test-instances目前均断裂，可能会将来被删除。

## 10.4 Basic Options Reference

以下部分仅概述基本选项

### 10.4.1 SMAC Options

运行SMAC的一般选项

#### BASIC OPTIONS

- help show help - 在此级别或更低的级别显示选项

默认值: 基本域: {基本, 中级, 高级, 开发人员}

- 当SMAC完成时, 执行验证

**Default Value:** true

**Domain:** {true, false}

-v打印版本和退出

### 10.4.2 Scenario Options

标准方案选项用于SMAC。一般认为使用-scenariofile指令指定这些参数和算法执行选项

#### BASIC OPTIONS

- 文件 - 文件包含所有实例的功能 - 包含要用于方案的实例的文件或目录。如果它是一个文件, 它必须符合a

具体格式(请参阅手册的实例文件格式部分), 如果它是它的目录, 您还必须使用--instance-exifix选项来限制匹配(除非所有文件都具有相同的扩展名), 并且实例列表将在排序订单。

#### REQUIRED

**Default Value:** null

- 每个目标算法的-run-obj运行我们最小化的目标类型

#### REQUIRED

**Default Value:** null

**Domain:** {RUNTIME, QUALITY}

--scenario-file scenario file

**Domain:** FILES

--skip - 功能如果为true将忽略功能文件(如果需要该功能文件, 则会导致错误, 好像没有提供

**Default Value:** false

**Domain:** {true, false}

- Stations文件或包含用于方案用于方案的实例。如果它是它必须coform的文件  
特定格式（请参阅手册的实例文件格式部分），如果它是目录，则必须使用--test-instance-suffix选项来限制匹配（除非所有文件具有相同的扩展名），以及实例列表将在排序订单默认值：null

#### 10.4.3 方案配置限制选项

控制方案将运行多长时间的选项

##### BASIC OPTIONS

- 电机限制限制SMAC和目标算法之间允许的总CPU时间运行  
自动配置阶段默认值：214748364  
7域：[0,2147483647]

- runcount-limit限制自动配置期间允许的目标算法总数

phase

**Default Value:** 9223372036854775807

**Domain:** (0, 9223372036854775807]

- 空间限制限制自动配置阶段期间允许的总壁钟时间

**Default Value:** 2147483647

**Domain:** (0, 2147483647]

#### 10.4.4 Algorithm Execution Options

与调用目标算法相关的选项

##### BASIC OPTIONS

- 单个目标算法运行的ALGO-CUTOFF-TIME CPU时间限制

**Default Value:** 1.7976931348623157E308

**Domain:** (0,  $\infty$ )

- Algo-Methalistic将目标算法视为确定性

**Default Value:** true

**Domain:** {true, false}

- ocg-exec命令字符串以执行算法

**REQUIRED**

**Default Value:** null

- 在PCS格式中包含包含算法参数空间信息的文件文件（参见算法参数文件中的文件）。您可以指定“Singleton”以获取单例配置空间或“null”以获得空值。必需的默认值：null

-t目标算法执行所需的其他上下文（参见可能值的TAE文档，  
generally rare)

**Default Value:**

## 10.5 Complete Options Reference

### 10.5.1 SMAC Options

运行SMAC的一般选项

#### BASIC OPTIONS

**--help** show help

**Aliases:** --help, -?, /?, -h

- Help级别显示此级别或更低的选项

别名: - help级默认值: 基本域: {基本, 中级, 高级, 开发人员}

- 当SMAC完成时, 执行验证

**Aliases:** --validation, --doValidation

**Default Value:** true

**Domain:** {true, false}

-v打印版本和退出

**Aliases:** -v, --version

#### INTERMEDIATE OPTIONS

- 适用 - 封盖使用自适应封盖

别名: --ATAPTIVE-CAPPAPPE, --AC, --ADAPTIVECAPPAPPE默认值: 默认为  
-runobj运行时为true, 否则为域: {true, false}

--always-run-initial-config如果为true, 我们将始终运行默认值并将其切换回它, 如果优于

现任别名: -

runway-run-initial-config, --alwaysRuninitialConfiguration默认值: 假域: {True, False}

--Console-log级默认日志级别控制台输出 (这不能比LogLevel更冗长)

别名: --Console-log-locl, --consolelevel默认值: 信息域: {跟踪, 调试, 信息, 警告, 错误, 关闭}

- DETERMINICINARIST-instancy-distracting如果为true, 则将从实例列表文件中选择实例

指定的订单别名: --deterministic-instancy-drounding, --deterministicinstandord默认值: false域: {true, false}

- Exec-Mode执行模式的自动配置器

**Aliases:** --exec-mode, --execution-mode, --executionMode

**Default Value:** SMAC

**Domain:** {SMAC, ROAR, PSEL}

- 实验文件夹的Dir Dir根目录

别名: - DIR, -  
experimentdir, -e默认值: <当前工作目录>

- 在挑战者上加强挑战者时, - 挑战 - 挑战者运行的初始持续金额

**Aliases:** --initial-challenger-runs, --initialN, --initialChallenge

**Default Value:** 1

**Domain:** (0, 2147483647]

- Initial-Dismumbent用于配置的初始现任现任 (您可以随机使用, 或默认为a

特殊字符串可随机或根据需要进行默认配置)。其他配置指定为: -Name'值'-name'value'...例如: --quick-sort'  
开'

**Aliases:** --initial-incumbent, --initialIncumbent

**Default Value:** DEFAULT

- Initial-Disbune - 运行初始运行量以计划默认配置

**Aliases:** --initial-incumbent-runs, --initialIncumbentRuns, --defaultConfigRuns

**Default Value:** 1

**Domain:** (0, 2147483647]

- 如果它们是严重性或更高, 则只会记录级别的消息。

别名: -  
Log-Level, --LogLevel默认值: Info域: {Trace, Debug, Info, War  
n, Error, Off}

- NUM运行的数量 (用于文件生成等)。这也控制了种子。

别名: --num-run, --numrun, --numrun, --sseed  
默认值: 随机生成的域: [0,2147483647]

- restore-scenario在状态文件夹中还原方案和状态

别名: - restore-scenario, -  
restorescenario默认值: null域: 文件

- rungroup outputdir子文件夹的名称, 以保存此运行的所有输出文件

**Aliases:** --rungroup, --rungroup-name, --runGroupName

**Default Value:**

- Save-runs-every-creation, 如果true将运行和结果文件保存到磁盘每次迭代。如果你的话有用

运行昂贵且您的群集不可靠, 如果您的运行很短, 则不必要地增加, 因为这可能增加了不可接受的开销量

**Aliases:** --save-runs-every-iteration

**Default Value:** false

**Domain:** {true, false}

- Show-hidding显示没有人使用的隐藏参数，并且可能只是打破smac（没有 - arguments）

**Aliases:** --show-hidden, --showHiddenParameters

- 验证时使用的核心数 - 仅在使用本地命令时适用

线路核心)。基本上，这会改变SMAC运行后的-Cli-Cores和-Cores的值。如果使用的目标是目标不是CLI，则使用此参数的使用是未定义的

别名: --Validation-Cores默认值: -Cor  
k域的值:( 0,2147483647]

- 野星的位置用于热启动

别名: - - - 从默认值--warmstart, - 从默认值: n /  
a（没有州是热烈的状态）

## ADVANCED OPTIONS

- AC-Add-Slack的数量增加了挑战者的计算自适应封盖价值（缩放后）

**Aliases:** --ac-add-slack, --capAddSlack

**Default Value:** 1.0

**Domain:** (0,  $\infty$ )

- Mult-Mull-Slack的数量以计算挑战者的计算自适应封盖价值

**Aliases:** --ac-mult-slack, --capSlack

**Default Value:** 1.3

**Domain:** (0,  $\infty$ )

- 用于在本地搜索期间使用的ACQ-Func采集功能

别名: - ACQ-Func, aacquistion-function, - func, - expected - 改进函数, -  
expectedimproventfunctionfunction默认值: 指数如果最小化运行时, 否则ei。域名: {指数, 简单, LCB, EI, LCBEIRR}

- 如果SMAC成功完成，则查询 - 旧式 - 成功将清除大部分无用状态文件

**Aliases:** --clean-old-state-on-success, --cleanOldStateOnSuccess

**Default Value:** true

**Domain:** {true, false}

- CONFIG跟踪采取配置的测量，因为它通过它的生命周期并写入文件（在 state folder）

**Aliases:** --config-tracking

**Domain:** {true, false}

- 覆盖 - 挑战挑战者与加倍封盖机制一起使用的挑战者数量

**Aliases:** --doubling-capping-challengers



**Default Value:** 2

**Domain:** (0, 2147483647]

- 覆盖覆盖的覆盖范围，每个挑战者的运行次数都会随着翻倍而得到的

盖上启动策略别名：- 螺旋覆盖 -

按挑战者默认值：2域：(0, 2147483647]

- help-default-file 文件，包含smac的默认设置

**Aliases:** --help-default-file, --helpDefaultsFile

**Default Value:** /.aeatk/help.opt

**Domain:** FILES

- 在构建模型时，截核 - 迭代次数以赋予审查数据的次数

**Aliases:** --imputation-iterations, --imputationIterations

**Default Value:** 2

**Domain:** [0, 2147483647]

## --init-mode Initialization Mode

别名：--init-mode, --initialization-mode, --initmode, --initializationMode默认值

: Classic Domain: {Classic, Iterative Capping, Unbiased Table}

- 花费加刷与模型学习的时间百分比的百分比百分比

**Aliases:** --intensification-percentage, --intensificationPercentage, --frac\_rawruntime

**Default Value:** 0.5

**Domain:** (0, 1)

- Intermediary-saves 确定是否制作任何中介省份（如果是假，则不快速保存

将要么）。状态仍将保存在运行的末尾但是别名：- -

保存默认值：true域：{true, false}

- 初始化阶段第2阶段中的tirtataTiveCappingBrefpletion，我们将中止第一个

完成时间的时间，而不是看其他任何kappa限制别名的其他东西

## --iterativeCappingK Iterative Capping K

**Aliases:** --iterativeCappingK

**Default Value:** 1

- 扫描 - 被审查的数据 - kappa-max面具被审查为kappa max的数据

**Aliases:** --mask-censored-data-as-kappa-max, --maskCensoredDataAsKappaMax

**Default Value:** false

**Domain:** {true, false}

- 扫描 - 非活动条件 - 参数 - 默认值构建模型处理非活动条件值作为默认值

**Aliases:** --mask-inactive-conditional-parameters-as-default-value, --maskInactiveConditionalParametersAsDefaultValue

**Default Value:** true

**Domain:** {true, false}

- max-dismunest-runs 允许的最大现有运行数量

**Aliases:** --max-incumbent-runs, --maxIncumbentRuns, --maxRunsForIncumbent

**Default Value:** 2000

**Domain:** (0, 2147483647]

- 伦 - 挑战者当地搜索所需的挑战人数

**Aliases:** --num-challengers, --numChallengers, --numberOfChallengers

**Default Value:** 10

**Domain:** (0, 2147483647]

- num-ei-walant of 随机配置, 以在EI搜索期间进行评估

**Aliases:** --num-ei-random, --numEIRandomConfigs, --numberOfRandomConfigsInEI, --numRandomConfigsInEI, --numberOfEIRandomConfigs

**Default Value:** 10000

**Domain:** [0, 2147483647]

- num-pca 在构建模型时使用的主要组件数量

**Aliases:** --num-pca, --numPCA

**Default Value:** 7

**Domain:** (0, 2147483647]

- 从文件中读取文件读取选项

**Aliases:** --option-file, --optionFile

**Domain:** FILES

- Option-file2 从文件中读取选项

**Aliases:** --option-file2, --optionFile2, --secondaryOptionsFile

**Domain:** FILES

- replertroupp-reply-and-exit 打印runggroupp中的所有可能的替换, 然后退出

**Aliases:** --print-runggroupp-replacement-and-exit

**Default Value:** false

**Domain:** {true, false}

- 正常保存确定是否快速保存

**Aliases:** --quick-saves

**Default Value:** true

**Domain:** {true, false}

- 迭代迭代状态要恢复，请使用“自动”自动选择最后一次迭代

别名: - 迭代 - - 迭代, - restorEttate符号, - restoreIteation默认值: n / a (没有恢复状态)

- restore-state-从状态的位置恢复

别名: - restore-state-from, - - restoreTatefrom默认值: n / a (没有恢复状态)

- save-ancnodut使用状态文件夹保存一些上下文，以便数据大多是自描述的（方案，实例文件，功能文件，保存Param文件）

**Aliases:** --save-context, --saveContext, --saveContextWithState

**Default Value:** true

**Domain:** {true, false}

- 分享模型 - 模式如果为true运行数据将被写入JSON文件和匹配特定文件的其他文件格式将在定期读取

别名: - 分享模型模式, --Share-Model-Mode, - 分享运行数据, - run-data默认值: 假域: {true, false}

- 模型模型 - 模式 - 频率如何调用新运行数据（以秒为单位）

**Aliases:** --shared-model-mode-frequency, --share-model-mode-frequency, --shared-run-data-frequency, --share-run-data-frequency

**Default Value:** 300 seconds

**Domain:** (0, 2147483647]

- smac-default-file文件，包含smac的默认设置

**Aliases:** --smac-default-file, --smacDefaultsFile

**Default Value:** /.aeatk/smac.opt

**Domain:** FILES

- state-deserializer确定将保存状态存储的文件的格式恢复

**Aliases:** --state-deserializer, --stateDeserializer

**Default Value:** LEGACY

**Domain:** {NULL, LEGACY}

- state-serializer确定要保存状态的文件的格式

**Aliases:** --state-serializer, --stateSerializer

**Default Value:** LEGACY

**Domain:** {NULL, LEGACY}

- Treat-Cunsoned-Data-As-Undensorion构建模型AS - 如果为帽值观察到响应值，是正确的[不推荐]

**Aliases:** --treat-censored-data-as-uncensored, --treatCensoredDataAsUncensored

**Default Value:** false

**Domain:** {true, false}

- 卧言 - 封顶 - 挑战者我们将在初始化期间考虑的挑战者的数量

**Aliases:** --unbiased-capping-challengers

**Default Value:** 2

**Domain:** (0, 2147483647]

- 禁止 - CPU3UPIT的CPU时间为初始化阶段花费构建表

**Aliases:** --unbiased-capping-cpulimit

**Default Value:** 0

**Domain:** (0, 2147483647]

- 卧际为止 - 覆盖的运行运行运行次数我们将在每个挑战期间考虑

lenger

**Aliases:** --unbiased-capping-runs-per-challenger

**Default Value:** 2

**Domain:** (0, 2147483647]

- 过验证种子用于验证SMAC

别名: - 验证 - 种子默认值: 0应该导致它与独立实用程序完全相同。

- 武术 - 迭代状态用于热启动, 使用“自动”自动挑选

last iteration

**Aliases:** --warmstart-iteration

**Default Value:** AUTO (if being restored)

## DEVELOPER OPTIONS

从种子中使用的numrun偏移偏移量 (这个加上的--numrun应该小于整数)

**Aliases:** --seed-offset, --seedOffset

**Default Value:** 0

- s在随机池中设置特定的种子 (按名称) (例如 - 柔道=2 -sInstance=4)。确定将使用的实际名称您应该使用启用调试日志记录运行程序, 结尾应该输出。

**Aliases:** -S

## 10.5.2 Random Forest Options

在构建随机林时使用的选项

### ADVANCED OPTIONS

- 完整树 - bootstrap引导将所有数据指向树

**Aliases:** --rf-full-tree-bootstrap, --fullTreeBootstrap

**Default Value:** false

**Domain:** {true, false}

- 忽略条件条件忽略构建模型的条件

**Aliases:** --rf-ignore-conditional, --ignoreConditionality

**Default Value:** false

**Domain:** {true, false}

- 赋予赋予均值为所有审查数据点的平均值施加

**Aliases:** --rf-impute-mean, --imputeMean

**Default Value:** false

**Domain:** {true, false}

--rf-log-model在日志正常形式中存储响应值

别名: --rf-log-model, --log-model, --logmodel默认值: true如果优化运行时, 如果优化质量域: {true, false}

**--rf-min-variance** minimum allowed variance

**Aliases:** --rf-min-variance, --minVariance

**Default Value:** 1.0E-14

**Domain:** (0,  $\infty$ )

- 在随机森林中创建树木的rnf-num-trees的数量

**Aliases:** --rf-num-trees, --num-trees, --numTrees, --nTrees, --numberOfTrees

**Default Value:** 10

**Domain:** (0, 2147483647]

- 惩罚算值算值处理占截止时间之上的算值, 低于惩罚

最大时间, 因为惩罚的最大时间

**Aliases:** --rf-penalize-imputed-values, --penalizeImputedValues

**Default Value:** false

**Domain:** {true, false}

- 拆分节点时, 可以考虑的功能数量的--rf比率 - 特征

**Aliases:** --rf-ratio-features, --ratioFeatures

**Default Value:** 0.8333333333333334

**Domain:** (0, 1]

--rf-shuffle-ust-value值随着树木之间的避免价值预测

**Aliases:** --rf-shuffle-imputed-values, --shuffleImputedValues

**Default Value:** false

**Domain:** {true, false}

- 拆分节点所需的最小元素数

**Aliases:** --rf-split-min, --split-min, --splitMin

**Default Value:** 10

**Domain:** [0, 2147483647]

#### DEVELOPER OPTIONS

- 预处理 - 边际建立随机森林, 具有预处理的边缘

**Aliases:** --rf-preprocess-marginal, preprocessMarginal

**Default Value:** true

**Domain:** {true, false}

- -RF-Store-Data Store在树叶中的完整数据

**Aliases:** --rf-store-data, --rf-store-data-in-leaves, --storeDataInLeaves

**Default Value:** false

**Domain:** {true, false}

- rf-subsample-memory-inction百分比, 当免费内存百分比下降低于此百分比我们将应用subsample百分比

**Aliases:** --rf-subsample-memory-percentage, --freeMemoryPercentageToSubsample

**Default Value:** 0.25

**Domain:** (0, 1]

- rf-subsample-inclipage乘以通过此值构建模型时使用的点数

**Aliases:** --rf-subsample-percentage, --subsamplePercentage

**Default Value:** 0.9

**Domain:** (0, 1]

- rf-subsample-values - 当MEM-数量的数量时, 输入值

ORY可用下降低于某个阈值 (参见 - subsamplevalueswhenlowmemory) (未测试)

**Aliases:** --rf-subsample-values-when-low-on-memory, --subsampleValuesWhenLowOnMemory, --subsampleValuesWhenLowMemory

**Default Value:** false

**Domain:** {true, false}

### 10.5.3 Scenario Options

标准方案选项用于SMAC。一般认为使用-scenariofile指令指定这些参数和算法执行选项

#### BASIC OPTIONS

- 文件文件文件, 包含所有实例功能

**Aliases:** --feature-file, --instanceFeatureFile, --feature\_file

- 签到文件或目录, 其中包含用于方案的实例。如果它是一个文件, 它必须coform a

具体格式 (请参阅手册的实例文件格式部分), 如果它是它的目录, 您还必须使用--instance-exifix选项来限制匹配 (除非所有文件都具有相同的扩展名), 并且实例列表将在排序订单。

必需的别名: --instances, --instance-file, --instance-dir, --instancefile, -i, --instance文件, -静音种子文件

**Default Value:** null

- 每个目标算法的`-run-obj`运行我们最小化的目标类型

必需的别名: `--run-obj`, `--run-object`, `--runobj`, `--run-obj`  
obj默认值: null域: {运行时, 质量}

**--scenario-file** scenario file

**Aliases:** `--scenario-file`, `--scenarioFile`, `--scenario`

**Domain:** FILES

- `--skip` - 功能如果为true将忽略功能文件 (如果需要该功能文件, 则会导致错误,

好像它没有提供别名: `--skip` -  
功能, `--ignore-itably`默认值: 假域: {true, false}

- Stations文件或包含用于方案用于方案的实例。如果它是它必须coform的文件

特定格式 (请参阅手册的实例文件格式部分), 如果它是目录, 则必须使用`--test-instance-suffix`选项来限制匹配 (除非所有文件具有相同的扩展名), 以及实例列表将在排序的秩序别名: `--test-`实例, `--test-instance-file`, `--test-instance-dir`, `- testinstancefile`, `--test`实例文件,

- 最低实例种子文件

**Default Value:** null

## INTERMEDIATE OPTIONS

- `--instance-upfix`, 所有实例必须在从目录中读取实例时匹配的后缀。你可以

(可选) 可选地指定正则表达式, 但请注意它是后缀匹配 (内部我们拍摄此字符串并追加为其) 别名: `--instance-suffix`, `--instance-Regex`默认值: null

- `--Intra-obj`目标函数用于聚合单个实例的多个运行

**Aliases:** `--intra-obj`, `--intra-instance-obj`, `--overall-obj`, `--intraInstanceObj`, `--overallObj`, `--overall_obj`, `--intra_instance_obj`

默认值: 意味着`-run-obj`是质量, 平均值如果它是运行时域: {均值, 均值, 平均值}。

**--output-dir** Output Directory

别名: `--output-dir`, `--outputdirectory`, `--outdir`默认值: <当前工作目录> / `-output`

- `Station-instance` - 后缀a后缀, 所有实例在从目录中读取实例时必须匹配。你

可以选择指定 (java) 正则表达式, 但请注意它是后缀匹配 (内部我们拍摄此字符串并追加它) 别名: `- 实例 - 后缀`, `--test-instance-regex`

**Default Value:** null

- 如果假跳过读取实例并使用虚拟实例，则 - 使用实例

**Aliases:** --use-instances

**Default Value:** true

**Domain:** {true, false}

#### ADVANCED OPTIONS

- 检查实例 - 存在检查磁盘上是否存在实例文件

**Aliases:** --check-instances-exist, --checkInstanceFilesExist

**Default Value:** false

**Domain:** {true, false}

- 用于聚合多个实例的Inter-obj客观函数 (已汇总

under the Intra-Instance Objective)

别名: --Inter-obj, --inter-instance-obj, --interinstanceobj, --inter instance  
obj默认值: 均值域: {mean, mean1000, mean10}

#### 10.5.4场景配置限制选项

控制方案将运行多长时间的选项

#### BASIC OPTIONS

- 电机限制限制SMAC和目标算法之间允许的总CPU时间运行

automatic configuration phase

**Aliases:** --cputime-limit, --cputime\_limit, --tunertime-limit, --tuner-timeout, --tunerTimeout

**Default Value:** 2147483647

**Domain:** [0, 2147483647]

- runcount-limit限制自动配置期间允许的目标算法总数

phase

**Aliases:** --runcount-limit, --runcount\_limit, --totalNumRunsLimit, --numRunsLimit, --numberOfRunsLimit

**Default Value:** 9223372036854775807

**Domain:** (0, 9223372036854775807]

- 空间限制限制自动配置阶段期间允许的总壁钟时间

**Aliases:** --wallclock-limit, --wallclock\_limit, --runtime-limit, --runtimeLimit, --wallClockLimit

**Default Value:** 2147483647

**Domain:** (0, 2147483647]

#### ADVANCED OPTIONS

- 符号限制限制自动配置阶段期间允许的迭代次数

**Aliases:** --iteration-limit, --numIterations, --numberOfIterations

**Default Value:** 2147483647

**Domain:** (0, 2147483647]



- 如果参数空间太小，则攻击 - 限制我们可能会到达我们可以做出的点  
没有新的跑步，检测到这种情况是非常昂贵的，这种启发式控制我们需要尝试挑战者的次数，并在我们放弃之前没有新的运行

**Aliases:** --max-norun-challenge-limit, --maxConsecutiveFailedChallengeIncumbent

**Default Value:** 1000

如果删除此文件，则删除终止终止该过程

**Aliases:** --terminate-on-delete

**Default Value:** null

--USE-CPU - Time-in-Tunertime包括SMAC的CPU时间作为TUNERTIMEOUT的一部分

**Aliases:** --use-cpu-time-in-tunertime, --countSMACTimeAsTunerTime

**Default Value:** true

**Domain:** {true, false}

## 10.5.5 Algorithm Execution Options

与调用目标算法相关的选项

### BASIC OPTIONS

- 单个目标算法运行的ALGO-CUTOFF-TIME CPU时间限制

别名: - -algo-cutoff-time, - -target-run-cuttime-limit, - - -target运行cpptime limit, - -cutoff-time, --cutoffTime, --cutoff\_time

**Default Value:** 1.7976931348623157E308

**Domain:**  $(0, \infty)$

- Algo-Methalistic将目标算法视为确定性

**Aliases:** --algo-deterministic, --deterministic

**Default Value:** true

**Domain:** {true, false}

- ocg-exec命令字符串以执行算法

### REQUIRED

**Aliases:** --algo-exec, --algoExec, --algo

**Default Value:** null

- 在PCS格式中包含包含算法参数空间信息的文件文件（参见算法参数文件中的文件）。您可以指定“Singleton”以获取单例配置空间或“null”以获得空值。

必需的别名: --pcs-file, --param文件, -p, --paramfile, --paramfile默认值: null

- t目标算法执行所需的其他上下文（参见可能值的TAE文档，generally rare)

**Aliases:** -T

### Default Value:

#### INTERMEDIATE OPTIONS

- ALGO-EXEC-DIR 工作目录以执行算法  
别名: --algo-exec-dir, --exec-dir, --execdir, --execdir默  
认值: 当前工作目录

#### ADVANCED OPTIONS

- 连续参数的邻居邻居数量  
**Aliases:** --continous-neighbours, --continuous-neighbors, --continuousNeighbours  
**Default Value:** 4

#### DEVELOPER OPTIONS

- search-subspace仅使用这些值生成随机和相邻配置。在A中指定  
“name = value, name = value, .....” 格式 (覆盖文件中设置的)

**Aliases:** --search-subspace, --searchSubspace  
**Default Value:** null

- search-subspace-file仅使用这些值生成随机和邻居配置。指定的  
每个自己的行上的每个参数都有个别值  
别名: --search-subspace-file, --searchsubspacefile默认值: n  
ull域: 文件

#### 10.5.6 目标算法评估器选项

描述和控制算法执行策略和机制的选项

#### INTERMEDIATE OPTIONS

- 崩溃的崩溃处理算法崩溃为中止 (如果算法不应该崩溃有用)。笔记:  
如果所有重试失败, 则只会中止。

**Aliases:** --abort-on-crash, --abortOnCrash  
**Default Value:** false  
**Domain:** {true, false}

- 如果算法的第一个运行崩溃将其视为中止, 则为首先碰撞崩溃  
allow crashes.

**Aliases:** --abort-on-first-run-crash, --abortOnFirstRunCrash  
**Default Value:** true  
**Domain:** {true, false}

- 运行 - 运行[已弃用] (使用TAE上的选项如果为true, 则仅允许--cors  
要同时进行评估的运行数。

**Aliases:** --bound-runs, --boundRuns  
**Default Value:** false  
**Domain:** {true, false}

--Check-SAT - 一致性确保在同一问题上运行实例始终返回相同的SAT / unsat result

**Aliases:** --check-sat-consistency, --checkSATConsistency

**Default Value:** true

**Domain:** {true, false}

--Check-SAT--一致性 - 异常抛出异常如果在同一问题上运行情况不同意 respect to SAT/UNSAT

**Aliases:** --check-sat-consistency-exception, --checkSATConsistencyException

**Default Value:** true

**Domain:** {true, false}

--cores [已弃用] (使用TAE选项, 如果可用) 最大并发目标数 algorithm executions

**Aliases:** --cores, --numConcurrentAlgoExecs, --maxConcurrentAlgoExecs, --numberOfConcurrentAlgoExecs

**Default Value:** 1

- run-run-forping-captime尝试杀死超出它们的运行超过一些金额

**Aliases:** --kill-run-exceeding-captime

**Default Value:** true

**Domain:** {true, false}

- 克里克利超越 - 克引力因素试图杀死超出这个因素的运行

**Aliases:** --kill-run-exceeding-captime-factor

**Default Value:** 10.0

**Domain:** (1,  $\infty$ )

- 重试算法在报告崩溃之前重试算法的次数 (注意: 原始崩溃不计入任何时间限制, 它们效果损失)。此外, 只有重试运行崩溃, 而不是中止运行, 这是按照设计, 因为中止只是在案例中我们不应该烦扰进一步运行

**Aliases:** --retry-crashed-count, --retryCrashedRunCount, --retryTargetAlgorithmRunCount

**Default Value:** 0

**Domain:** [0, 2147483647]

--TAE目标算法评估器在制作目标算法时使用

**Aliases:** --tae, --targetAlgorithmEvaluator

**Default Value:** CLI

**Domain:** {ANALYTIC, BLACKHOLE, CLI, CONSTANT, IPC, PRELOADED, RANDOM}

- 如果true输出输出目录中的文件, 则会运行--track-scheduld-运行 在任何给定的时间评估

**Aliases:** --track-scheduled-runs

**Default Value:** false

**Domain:** {true, false}

- verify-sat检查算法的SAT / unsat / 未知响应，其中值存储为实例

特定信息，如果存在差异，请记录错误。基于每个问题实例的实例特定信息的实例的值自动检测默认值。如果每个实例都有一个实例特定信息，请在以下设置sat，unsat，未知，满足，不满意，这将设置为true，否则它将是假的。别名：- Verify-SAT，- Verify-SAT，- VerifySAT默认值：自动检测到（参见描述）域：{true，false}

## ADVANCED OPTIONS

- 在完成之前，可以在完成之前确保在完成之前调用TAE观察者

**Aliases:** --call-observer-before-completion

**Default Value:** true

**Domain:** {true, false}

- 文件 - 缓存如果True运行将从指定的输入和输出文件中写入或读取。如果是目录

指定，然后输入将来自目录中的所有文件，并且输出将在目录中的新随机文件。注意：此缓存是静态的，我们不会通过时间别名从缓存重新读取：--file-cache默认值：false域：{true，false}

- 文件 - 缓存 - 输出从哪里写入文件

**Aliases:** --file-cache-output

**Default Value:** null

- 文件 - 缓存源从中读取文件

**Aliases:** --file-cache-source

**Default Value:** null

- --log-requests-ancestes如果设置为true，所有评估请求将在提交时记录和

已完成的别名：--log-requests-responses默认值：false域：{true，false}

- --log-requests-responses-rc-only，仅当设置为true时，我们只会在运行完成时记录运行配置

**Aliases:** --log-requests-responses-rc-only, --log-requests-responses-rc

**Default Value:** false

**Domain:** {true, false}

- 在我们使用之前，波索瓦 - Walltime-延迟等待使用运行时信息的更新需要多长时间

Walltime。5秒和0.95的等级，意味着我们将看到0,0,0,0 ..., 4.95

...别名：--observer-walltime延迟默认值：5.0域：( 0, ∞ )

- 如果是true和目标算法，如果true和target算法与运行时的信息不更新我们 -

Mation

We报告墙壁时间别名: --observer-walltime-if-of-runtime默认值: true域: {true, false}

- 波持程序 - Walltime-Scale WallTime应该用作运行时的哪些因素 (通常会推荐 -

修补程序是核心数量的0.95倍) 别名: - 波持程序

- Walltime-Scale默认值: 0.95域: (0,  $\infty$ )

- 包含目标算法评估符的默认设置的 - 默认文件文件

**Aliases:** --tae-default-file

**Default Value:** /.aeatk/tae.opt

**Domain:** FILES

--track-scheduld-runs-解决方案我们将桶更改为此大小

**Aliases:** --track-scheduled-runs-resolution

**Default Value:** 1.0

**Domain:** (0,  $\infty$ )

## DEVELOPER OPTIONS

--Cache运行 (如果为True, 我们将缓存在内部运行, 以便未重新执行后续请求[emper-

IMENTAL]

**Aliases:** --cache-runs

**Default Value:** false

**Domain:** {true, false}

- cache-runs-debug如果true, 我们将经常打印每种缓存的状态, 经常用于调试目的。

**Aliases:** --cache-runs-debug

**Default Value:** false

**Domain:** {true, false}

--Cache运行 - 严格上升观察者如果是的, 那么我们将强制执行外部看到的所有运行时

总是严格增加时间。(由于某种原因重新开始运行, 在内部, 观察时间实际上可能会下降)。别名: --Cache运行 - 严格上升观察者默认值: 假域: {true, false}

--Check-for-Unclean-shutdown如果为true, 我们将尝试检测目标算法的不洁关闭

评估仪别名: --Check

for-Unclean-shutdown默认值: True

Domain: {True, False}

--Check-for-unique-runconfigs检查批处理中的所有提交运行配置都是唯一的

**Aliases:** --check-for-unique-runconfigs

**Default Value:** true

**Domain:** {true, false}

--Check-for-unique-runconfigs-exception (如果为true), 我们将抛出异常, 如果重复运行configurations are detected

**Aliases:** --check-for-unique-runconfigs-exception

**Default Value:** true

**Domain:** {true, false}

- 检查结果 - 顺序 - 一致检查TAE以正确的顺序返回响应

**Aliases:** --check-result-order-consistent, --checkResultOrderConsistent

**Default Value:** false

**Domain:** {true, false}

--exception-on-prepost-command Throw an abort

**Aliases:** --exception-on-prepost-command, --exceptionOnPrePostCommand

**Domain:** {true, false}

- on-on-fally如果为true, 则检测到故障时, 该过程将尽力关闭, 可能不是cleanly

**Aliases:** --exit-on-failure

**Default Value:** false

**Domain:** {true, false}

- 文件 - 缓存 - 崩溃的缓存密度错误应用程序将在缓存未命中崩溃, 这是为了调试

别名: - 文件 - 缓存 - 崩溃 -

on-cache-miss, --file-cache-crash-on-more默认值: false域: {true, false}

- 如果删除该文件, 删除文件 - 删除文件 - 删除所有运行将被强制损失。此选项可能导致

应用程序进入无限循环如果删除文件, 则需要小心。通常, 您需要设置此问题和一些其他选项要指向同一文件, 如果还有另一个选项, 则应用程序可能会很好地关闭, 如果没有, 那么它可能是无限循环。

**Aliases:** --kill-runs-on-file-delete

**Default Value:** null

-post-scenario-command命令将在关机上运行

**Aliases:** --post-scenario-command, --postScenarioCommand, --post\_cmd

-pre-scenario-command命令将在启动时运行

**Aliases:** --pre-scenario-command, --preScenarioCommand, --pre\_cmd

- 用于预/帖子命令的verpost-exec-dir执行目录

**Aliases:** --prepost-exec-dir, --prePostExecDir

默认值: 当前工作目录域: {readBabledirectories}

--PREPOST-LOG-输出从预先和发布命令的所有输出记录

**Aliases:** --prepost-log-output, --logOutput

**Domain:** {true, false}

- 哈希码文件文件, 包含每行Run Hashes列表: 每行应该是: “运行哈希代码:

(哈希代码) (n) 运行“。此文件中的运行数量不需要与执行的运行数量不匹配, 此文件仅可确保序列从未发出过。注意N是完全忽略的, 所以它们所指定的顺序是我们预期此版本中哈希代码的顺序。最后注意您可以简单地指出以前的日志, 其他线路将被忽略

**Aliases:** --run-hashcode-file, --runHashCodeFile

**Domain:** FILES

- Skip-Operenge-Eval-TAE如果设置为真正的代码, TAE将不会被装饰者包裹以支持等待出色的跑步

**Aliases:** --skip-outstanding-eval-tae

**Default Value:** false

**Domain:** {true, false}

--tae-stop-procession-on-shutdown如果为true, 则一旦JVM关闭, 就在Appli中触发

阳离子或外部所有进一步的请求将被默默地丢弃。建议使用这一点, 因为否则应用可能会看到意外结果, 因为TAE可能无法继续处理。

**Aliases:** --tae-stop-processing-on-shutdown

**Default Value:** true

**Domain:** {true, false}

- tae-warn-if-no-respon-from-tae如果大于0, 则等待TAE的秒数  
在发出警告之前回复

**Aliases:** --tae-warn-if-no-response-from-tae

**Default Value:** 120

**Domain:** [0, 2147483647]

- 使用 - 动态截止如果是, 则我们将所有截止物更改为最大截止时间并动态杀死运行超过截止时间。这很有用, 因为缓存命中需要截止时间匹配

**Aliases:** --use-dynamic-cutoffs

**Default Value:** false

**Domain:** {true, false}

#### 10.5.7变换目标算法评估器装饰器选项

此目标算法评估器装饰器允许您根据某些规则转换包装器的响应值。可以由Exp4J (<http://www.objecthunter.net/exp4j/>) 使用的表达式, 并将导致返回的运行相应地转换。表达式中的变量可以是-1如果运行undat, 1, 如果sat, 则为0, 否则是原始报告的

运行时，Q是原始报告的质量，以及C是所要求的截止时间。在将值转换为遵守包装语义时，应注意。如果您不知道您在做什么，我们建议您的SAT和unsat值保持在0和截止之间的范围内，并且根本不应转换超时值。非常特别感谢原作者亚历山大FR'Echette。

## ADVANCED OPTIONS

- 如果您想转换结果，则tae-transform设置为true，如果false其他转换没有效果

**Aliases:** --tae-transform

**Default Value:** false.

**Domain:** {true, false}

- 如果结果是SAT，则将 - 转换 - SAT - 质量功能适用于算法运行的质量。

别名: - TAE-Transform-Sat - 质量默认值: Identity

Transport。域: 使用运行相关的变量的可计算字符串: s运行结果 (sat = 1, unsat = -1, 其他= 0),

## R runtime, Q quality, C cutoff.

- 如果结果SAT，则将 - 转换 - SAT运行时函数应用于算法运行的运行时。

别名: --tae-transform-sat-truntime默认值: 身份变换。

域: 使用运行相关的变量的可计算字符串: s运行结果 (sat = 1, unsat = -1, 其他= 0),

## R runtime, Q quality, C cutoff.

- 如果结果是超时，则将 - 转换 - 超时 - 质量函数应用于算法运行的质量。

别名: - TAE-Transform-Timeout - 质量默认值: Identity

Transport。域: 使用运行相关的变量的可计算字符串: s运行结果 (sat = 1, unsat = -1, 其他= 0),

## R runtime, Q quality, C cutoff.

- 如果结果是时间，则将转换 - 超时 - 运行时函数应用于算法运行的运行时 -

OUT.

别名: - tae-transform-timeout -

运行时默认值: 身份变换。域: 使用运行相关的变量的可计算字符串: s运行结果 (sat = 1, unsat = -1, 其他= 0),

## R runtime, Q quality, C cutoff.

- 如果结果unsat，则算法运行的算法运行质量的TAE-Transform-Transtat - Unsat ustat函数。

别名: - TAE-Transform-unsat Quest默认值: Identity

Transport。域: 使用运行相关的变量的可计算字符串: s运行结果 (sat = 1, unsat = -1, 其他= 0),

## R runtime, Q quality, C cutoff.

- 如果结果unsat，则将算法运行的运行时间应用于算法运行的unstat运行时功能。

**Aliases:** --tae-transform-UNSAT-runtime

**Default Value:** Identity transform.



域：使用运行相关的变量的可计算字符串：s运行结果（sat = 1，unsat = -1，其他= 0），

R runtime, Q quality, C cutoff.

- 如果结果不饱和，则申请算法运行的实质功能

or TIMEOUT.

别名：- TAE-变换 - 其他质量默认值：Identity

Transport。域：使用运行相关的变量的可计算字符串：s运行结果（sat = 1，unsat = -1，其他= 0），

R runtime, Q quality, C cutoff.

- 如果结果不饱和，则将转换 - 其他运行时函数应用于算法运行的运行时，

UNSAT or TIMEOUT.

别名：- TAE-变换 - 其他运行时默认值：Identity

Transport。域：使用运行相关的变量的可计算字符串：s运行结果（sat = 1，unsat = -1，其他= 0），

R runtime, Q quality, C cutoff.

## DEVELOPER OPTIONS

- tae-transform-有效值 - 仅当运行时的转换导致一个太大的值，即  
截止时间将返回，结果更改为超时。如果结果太小，则将设置为0

**Aliases:** --tae-transform-valid-values-only

**Default Value:** true

**Domain:** {true, false}

### 10.5.8分叉目标算法评估器装饰器选项

此目标算法评估器装饰员允许您委派某些运行到另一个TAE，表示奴隶TAE。实施了几项政策（或者将在请求/需要时）。前两项复制了从奴隶上的运行，主要动机是非常短的运行的性能，其中向主要派遣的开销可能会令人惊讶。接下来的两个（要实现），将允许某些操作以在主站或主机之后由从站完成。

## ADVANCED OPTIONS

- fork-to-tae如果不是null，运行也将同时提交给另一个TAE。第一个跆拳道

返回使用答案。

别名：--fork-to-tae默认值：禁用请求的分支是禁用的域：{分析，黑洞，CLI，常数，IPC，预加载，随机}

- fforc-to-to-tup-on-slave-quick-timeout当快速上的副本时使用了什么时间使用  
policy.

**Aliases:** --fork-to-tae-duplicate-on-slave-quick-timeout

**Default Value:** 5 seconds

**Domain:** (0, 2147483647]

--fork-to-tae-policy选择我们将与之叉的策略。例如，从奴隶上重复简单地提交运行到从属奴隶。从奴隶快速重复，将运行提交到奴隶，但截止时间减少

别名: --fork-to-tae-policy默认值: 如果forktotae不是null域, 则必须明确设置: {在从站上复制, 在从站上重复}

## 10.5.9 Validation Options

控制验证的选项

### INTERMEDIATE OPTIONS

- MAX-TIMESTAMP最大轨迹文件中的相对时间戳来配置。-1手段

auto-detect

别名: --max-timestamp, --maxtimestamp默认值

: 自动检测域:  $[0, \infty)$  {-1}

- Minuestamp轨迹文件中的最小相对时间戳来配置。

**Default Value:** 0.0

**Domain:**  $[0, \infty)$

--num验证 - 运行近似验证数量运行

**Aliases:** --num-validation-runs, --numValidationRuns, --numberOfValidationRuns

**Default Value:** 1

**Domain:**  $[0, 2147483647]$

--save-state-file保存由我们所做的所有运行组成的状态文件

**Aliases:** --save-state-file, --saveStateFile

**Default Value:** false

**Domain:** {true, false}

- 逐个墙壁验证时间验证挂钟时间

**Aliases:** --validate-by-wallclock-time, --validateByWallClockTime

**Default Value:** true

**Domain:** {true, false}

- 如果轨迹文件中的Walltime未达到此条目, 则达到仅限If-Tunertime-Tunertime-Tunertime-Tunertime-ant  
bother validating

**Aliases:** --validate-only-if-tunertime-reached, --validateOnlyIfTunerTimeReached

**Default Value:** 0.0

**Domain:**  $[0, \infty)$

- 如果轨迹文件中的WallTime没有按此条目, 我们不会达到才能达到-Worm-walltime-walltime-where-walltime-walltime-walltime-wall  
bother validating

**Aliases:** --validate-only-if-walltime-reached, --validateOnlyIfWallTimeReached

**Default Value:** 0.0

**Domain:**  $[0, \infty)$

– 仅限实际上现任现任实际验证只发现最后的现任者

**Aliases:** --validate-only-last-incumbent, --validateOnlyLastIncumbent

**Default Value:** true

**Domain:** {true, false}

## ADVANCED OPTIONS

– 时间戳的几何进度的Mult因数基础验证（例如默认情况下）

maxTimestamp, maxTimestamp/2, maxTimestamp/4,... while timestamp  $\geq$  minTimestamp )

**Aliases:** --mult-factor, --multFactor

**Default Value:** 2.0

**Domain:**  $(0, \infty)$

--output-file-suffix 后缀添加到验证运行文件（用于分组）

**Aliases:** --output-file-suffix, --outputFileSuffix

--validate-all 验证轨迹文件中的每个条目（覆盖其他验证选项）

**Aliases:** --validate-all, --validateAll

**Default Value:** false

**Domain:** {true, false}

--validation-rounding-mode 选择是否循环验证数量（到下一个倍数

numTestInstances

**Aliases:** --validation-rounding-mode, --validationRoundingMode

**Default Value:** UP

**Domain:** {UP, NONE}

## DEVELOPER OPTIONS

--Num-Seeds-rese-instance 弃用/破坏：在有效期间每隔使用的测试种子数量 –

tion

**Aliases:** --num-seeds-per-test-instance, --numSeedsPerTestInstance, --numberOfSeedsPerTestInstance

**Default Value:** 1000

**Domain:**  $(0, 2147483647]$

--num-test-instances 弃用/损坏：仔细检查结果：对抗的情况数量（将

执行此目的的min，以及测试实例文件中的实例数）。要禁用SMAC中的验证请参见--doporationation选项别名：  
--num-test-sirtances, --numtestinstances, --numberofftestinstances默认值： $(0, 2147483647]$

--vertiation-headers 将标题放在输出CSV文件上进行验证

**Aliases:** --validation-headers, --validationHeaders

**Default Value:** true

**Domain:** {true, false}

#### 10.5.10分析目标算法评估器选项

此目标算法评估器使用分析功能来生成运行时。大多数的努力 -

由Marcin Molga, Czesaw

Smutnicki ( <http://www.zsd.ic.pwr.wloc.pl/files/docs/functions.pdf> ) 来源来自测试功能的测试功能。注意: 某些功能已垂直移位, 以便响应值始终为正数。

#### ADVANCED OPTIONS

- 分析 - 使用哪些分析功能使用

别名: - 分析函数默认值: Camelback

Domain: {Zero, Add, Camelback, Branins}

#### DEVELOPER OPTIONS

- 分析观察者 - 频率如何通知观察者更新观察者 ( 以毫秒为单位 )

**Aliases:** --analytic-observer-frequency

**Default Value:** 100

**Domain:** (0, 2147483647]

- 分析 - 尺度 - 模拟 - 延迟延迟除以该值的模拟延迟

**Aliases:** --analytic-scale-simulate-delay

**Default Value:** 1.0

**Domain:** (0,  $\infty$ )

- 分析 - 模拟 - 核心如果设置为大于0, 则TAE将序列化请求, 以便不超过这些数字将同时执行。

**Aliases:** --analytic-simulate-cores

**Default Value:** 0

**Domain:** [0, 2147483647]

- 分析 - 模拟 - 如果设置为真, 则TAE将模拟壁板延迟

**Aliases:** --analytic-simulate-delay

**Default Value:** false

**Domain:** {true, false}

#### 10.5.11黑洞目标算法评估器选项

此目标算法评估程序根本从不返回任何运行

#### DEVELOPER OPTIONS

- blackhole-warnings抑制生成的警告

**Aliases:** --blackhole-warnings

**Default Value:** true

**Domain:** {true, false}

## 10.5.12 命令行目标算法评估器选项

该目标算法评估器通过命令行和标准包装器接口执行命令。

### INTERMEDIATE OPTIONS

- CLI - 并发执行是否允许并发执行

**Aliases:** --cli-concurrent-execution

**Default Value:** true

**Domain:** {true, false}

- 用于执行运行的Cli-Cores数量。换句话说，在给定的情况下运行的请求次数  
time.

**Aliases:** --cli-cores

**Default Value:** 1

**Domain:** (0, 2147483647]

- CLI-log-all-call-strings日志每呼叫字符串

**Aliases:** --cli-log-all-call-strings, --log-all-call-strings, --logAllCallStrings

**Default Value:** false

**Domain:** {true, false}

- CLI-log-all-call记录所有呼叫字符串和结果行

**Aliases:** --cli-log-all-calls, --cli-log-all-call-strings-and-results, --log-all-calls, --log-all-call-strings-and-results

**Default Value:** false

**Domain:** {true, false}

- CLI-log-All-Process-Output日志所有进程输出

**Aliases:** --cli-log-all-process-output, --log-all-process-output, --logAllProcessOutput

**Default Value:** false

**Domain:** {true, false}

- CLI-log-all-excues记录所有结果行

别名: -

CLI-log-all-结果, --cli-log-all-call-excues, --log-all-call-excuse, --log-all-结果默认值: false域  
: {true, 错误的}

### ADVANCED OPTIONS

- CLI-CALL-PARAMS带引号，如果对目标算法的真实呼叫有引用的参数  
“3” 而不是 “3”。代码的旧版本通过了 “参数”。这已被删除，将来将被推迟

**Aliases:** --cli-call-params-with-quotes

**Default Value:** false

**Domain:** {true, false}

- 包含CLI目标算法评估器的默认设置的CLLI-DEFAULT-FILE文件（建议使用  
您使用此文件来设置kill命令）

**Aliases:** --cli-default-file

**Default Value:** /.aeatk/cli-tae.opt

**Domain:** FILES

- CLI-KILL-BY-ENVIRONMENT-CMD如果不是null，则此脚本将用三个参数执行，第一个键，  
第二个价值，第三个是我们在PID的最佳猜测（-1意味着我们无法猜到）。它们表示环境名称和价值，脚本应查找  
每个名称和价值设置的每个进程并终止它。不要假设密钥是静态的，因为它可能会根据现有环境变量而改变。示例  
脚本可以在示例脚本/ env kill /

**Aliases:** --cli-kill-by-environment-cmd

**Default Value:** null

- CLI-LOATE-for-更新，如果TRUE将创建套接字并设置环境变量，以便我们可以拥有  
CPU时间的更新

**Aliases:** --cli-listen-for-updates

**Default Value:** true

**Domain:** {true, false}

- cli-pg-force-kill-cmd命令要尝试尝试并询问进程组终止很好（一般

a SIGKILL in Unix). Note

别名: --cli-pg-force-kill-cmd默认值: ba  
sh -c "kill -s kill -

- cli-pg-nice-kill-cmd命令执行尝试并要求进程组终止很好（一般

a SIGTERM in Unix). Note

别名: --cli-pg-nice-kill-cmd默认值: bash  
-c "kill -s术语 -

- cli-proc-force-kill-cmd命令执行尝试并要求进程终止（通常是一个

SIGTERM in Unix). Note

别名: --cli-proc-force-kill-cmd默认  
值: kill -s kill

- cli-proc-nice-kill-cmd命令执行尝试并要求进程终止终止（通常是一个

SIGTERM in Unix). Note

别名: --cli-proc-nice-kill-cmd默认  
值: kill -s术语

## DEVELOPER OPTIONS

- CLI观察者 - 频率通知观察者更新观察者的频率（以毫秒为单位）

**Aliases:** --cli-observer-frequency

**Default Value:** 500

**Domain:** (0, 2147483647]

### 10.5.13 常量目标算法评估器选项

常量目标算法评估器的参数

#### DEVELOPER OPTIONS

– 循环 – 额外的运行数据其他运行数据以返回

**Aliases:** --constant-additional-run-data

**--constant-run-length** Runlength to return

**Aliases:** --constant-run-length

**Default Value:** 0.0

– 普通质量质量返回

**Aliases:** --constant-run-quality

**Default Value:** 0.0

– runst-rul-rulity 运行结果返回

别名: – 旋转结果默认值: SAT域: {Timeout, SAT, unsat, 崩溃, 中止, 运行, 杀死}

**--constant-runtime** Runtime to return

**Aliases:** --constant-runtime

**Default Value:** 1.0

### 10.5.14 流程间通信目标算法评估器选项

该目标算法评估器将请求放在另一个过程中。当前编码机制与命令行相同，除了我们未指定Algo可执行字段。当前机制只能一次向服务器执行一个请求。需要小的代码更改来处理更常规的情况，因此如果需要，请联系开发人员。

#### ADVANCED OPTIONS

--ipc-async-threads 使用异步线程的数量

别名: --ipc-async-threads 默认值: 超过可用处理器的数量

– ipc-default-file 文件，包含IPC目标算法评估器的默认设置（建议您使用此文件来设置kill命令）

**Aliases:** --ipc-default-file

**Default Value:** /.aeatk/ipc-tae.opt

**Domain:** FILES

– 编码邮件如何编码

别名: –

编码默认值: 调用串域: {呼叫字符串, java序列化}

--ipc-exec-on-start-up此脚本将在IPC TAE的启动时执行。最后的论点是如果我们的IPCMEchaismism是反向TCP，则附加到服务器端口

**Aliases:** --ipc-exec-on-start-up, --ipc-exec

**Default Value:** null

--ipc-exec-eputn，如果为true，我们将从脚本中记录所有输出

**Aliases:** --ipc-exec-output

**Default Value:** false

**Domain:** {true, false}

--ipc-local-port本地服务器端口为某种类型的IPC机制（如果为0，则会自动由操作系统分配）

**Aliases:** --ipc-local-port

**Default Value:** 0

**Domain:** [1,65535]

- ipc用于使用的机制机制

**Aliases:** --ipc-mechanism

**Default Value:** UDP

**Domain:** {UDP, TCP, REVERSE\_TCP}

- 用于某些类型的IPC机制的 - 远程主机远程主机

**Aliases:** --ipc-remote-host

**Default Value:** 127.0.0.1

- iip-remote-port远程端口为某种类型的IPC机制

**Aliases:** --ipc-remote-port

**Default Value:** 5050

**Domain:** [0,65535]

- iipc-report-persistent是应该被视为持续存在的，如果我们的话，松散地是持久的可以稍后要求它同样的请求，并且不必重新从头开始重做工作。

**Aliases:** --ipc-report-persistent

**Default Value:** false

**Domain:** {true, false}

- iipc-reverse-tcp-pool连接如果为true，我们将汇集所有连接而不是关闭它们

**Aliases:** --ipc-reverse-tcp-pool-connections

**Default Value:** false

**Domain:** {true, false}

- iipc-udp-packetsize用于某些类型的IPC机制的远程端口

**Aliases:** --ipc-udp-packetsize

**Default Value:** 4096

**Domain:** [0,65535]



### 10.5.15预加载响应目标算法评估器

目标算法评估器提供预加载的响应

#### DEVELOPER OPTIONS

- 重新加载 - 额外的运行数据其他运行数据以返回

**Aliases:** --preload-additional-run-data

- 重载质量质量以返回所有值

**Aliases:** --preload-quality

**Default Value:** 0.0

- 重新加载 - 响应 - 数据预加载的响应值格式[sat, unsat, ... = x], 其中x是a runtime (e.g. [SAT=1],[UNSAT=1.1]...

**Aliases:** --preload-response-data, --preload-responseData

- 重新加载 - 运行长度runlength以返回所有值

**Aliases:** --preload-run-length, --preload-runLength

**Default Value:** -1.0

### 10.5.16随机目标算法评估器选项

该目标算法评估器随机地从均匀分布生成响应

#### DEVELOPER OPTIONS

- random-额外的运行数据其他运行数据以返回

**Aliases:** --random-additional-run-data

- random-max-响应我们将生成的最大运行时

**Aliases:** --random-max-response

**Default Value:** 10.0

**Domain:**  $[0, \infty)$

- random-min-response我们将生成的最小运行时间 (比0.01小于0.01的值 to 0.01)

**Aliases:** --random-min-response

**Default Value:** 0.0

**Domain:**  $[0, \infty)$

- random-Observer-频率如何通知观察者更新观察者 (以毫秒为单位)

**Aliases:** --random-observer-frequency

**Default Value:** 500

**Domain:** (0, 2147483647]

- 样品 - 种子种子在生成随机响应时使用

别名: - 样本 -

样本默认值: 当前时间以毫秒为单位

- 绘制尺度模拟 - 延迟延迟通过此值划分模拟延迟

**Aliases:** --random-scale-simulate-delay

**Default Value:** 1.0

**Domain:**  $(0, \infty)$

- random-simulate-cores如果设置为大于0，TAE将序列化请求，以便不超过这些数字将同时执行。

**Aliases:** --random-simulate-cores

**Default Value:** 0

**Domain:**  $[0, 2147483647]$

- random-simulate-delay如果设置为true，则TAE将模拟壁锁延迟

**Aliases:** --random-simulate-delay

**Default Value:** false

**Domain:** {true, false}

- 趋势 - 趋势系数第n个样本将从 $\text{Max}(0, \text{均匀}(\text{min}, \text{max}) + n \times (\text{趋势} - \text{系数}))$ 分布。这允许您随着时间的推移增加或减少响应值。

**Aliases:** --random-trend-coefficient

**Default Value:** 0.0