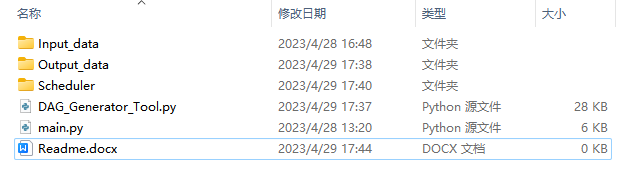
## 代码架构介绍



1. Input\_data文件夹

该文件夹包含DAG输入的相关数据



其中共有三类样例文件：

1）DAG\_data文件夹里的csv文件是一个完整的DAG文件；

2）DAG\_param文件夹里的xlsx文件是对一个DAG的抽象文件，文件中的数据不是完整的DAG，只是DAG抽象后的相关特性数据；

3）Flow\_data文件夹中的xlsx文件是一组完整的flow文件；

（2）Output\_data文件夹

代码生成数据的输出地址。

（3）Scheduler文件夹：

该文件夹中存储各种功能模块代码，介绍如下：

表1 代码列表及简介

|  |  |  |
| --- | --- | --- |
| ID | 文件名 | 简介 |
| 1 | Core.py | 仿真器使用的核模块 |
| 2 | DAG\_Data\_Processing.py | DAG数据处理模块 |
| 3 | DAG\_Features\_Analysis.py | DAG特性提取模块 |
| 4 | DAG\_Generator.py | DAG输入与DAG生成器模块 |
| 5 | DAG\_Priority\_Config.py | DAG配置各优先级 |
| 6 | DAG\_WCET\_Config.py | DAG生成配置随机执行时间 |
| 7 | limited\_priority\_assign.py | 有限优先级模块 |
| 8 | Scheduling\_Simulator.py | 仿真器模块 |
| 9 | Simulation\_Result\_Show.py | 实验结果展示用模块 |

（4）DAG\_Generator\_Tool.py文件：

DAG生成器代码，具体内容见章1（DAG生成工具）。

（5）main.py文件：

DAG调度仿真器代码，具体内容见章2（DAG仿真器）。

## DAG生成工具

### 1.1 DAG生成方式1-基于输入DAG生成

|  |  |
| --- | --- |
|  |  |
| 1. 基于输入DAG生成的基本步骤 | 1. 输入DAG文件后关键参数自动生成 |
|  |  |
| 1. 确定生成DAG的数量以及输出地址 | 1. 点击启动生成算法 |

具体步骤入下

Step1. 点击DAG参数抽象按钮，确定输入DAG数据的地址（输入数据的样例文件在.\Input\_data\DAG\_data中）；输入数据后，将自动生成DAG关键特性列表；

Step2. 在生成DAG数量编辑框中输入要生成的DAG数量，并点击打开按钮在输出\_Excal\_DAG\_Data\_文件路径中确定输出文件的地址；

Step3. 点击基于参数生成DAG按钮（中间会有一段时间的延迟）。完成后输出成功对话框。

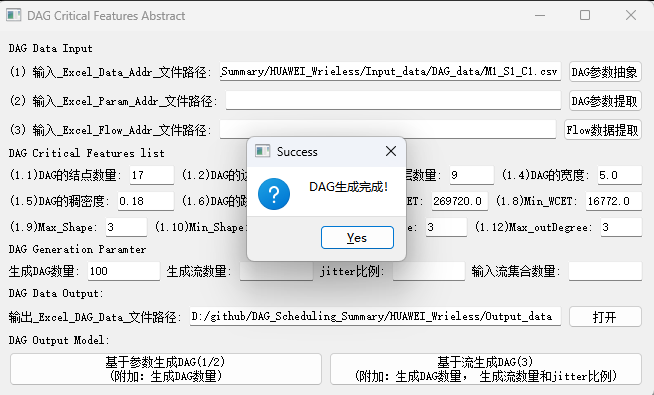


图 DAG生成工作完成

### 1.2 DAG生成方式2-基于输入关键参数生成

|  |  |
| --- | --- |
|  |  |
| 1. 输入DAG文件后关键参数自动生成 | 1. 确定生成DAG的数量以及输出地址 |
|  |  |
| (c) 点击启动生成算法 | (d) DAG生成工作完成 |

具体步骤入下

Step1. 点击DAG参数提取按钮，确定输入DAG特性数据的地址（输入数据的样例文件在.\Input\_data\DAG\_param中）；输入数据后，将自动生成DAG关键特性列表；

Step2. 在生成DAG数量编辑框中输入要生成的DAG数量，并点击打开按钮在输出\_Excal\_DAG\_Data\_文件路径中确定输出文件的地址；

Step3. 点击基于参数生成DAG按钮（中间会有一段时间的延迟）。完成后输出成功对话框。

### 1.3 DAG生成方式3-基于流的动态生成

|  |  |
| --- | --- |
|  |  |
| 输入DAG文件后关键参数自动生成 | 确定生成DAG的数量以及输出地址 |
|  |  |
| 点击启动生成算法 | DAG生成工作完成 |

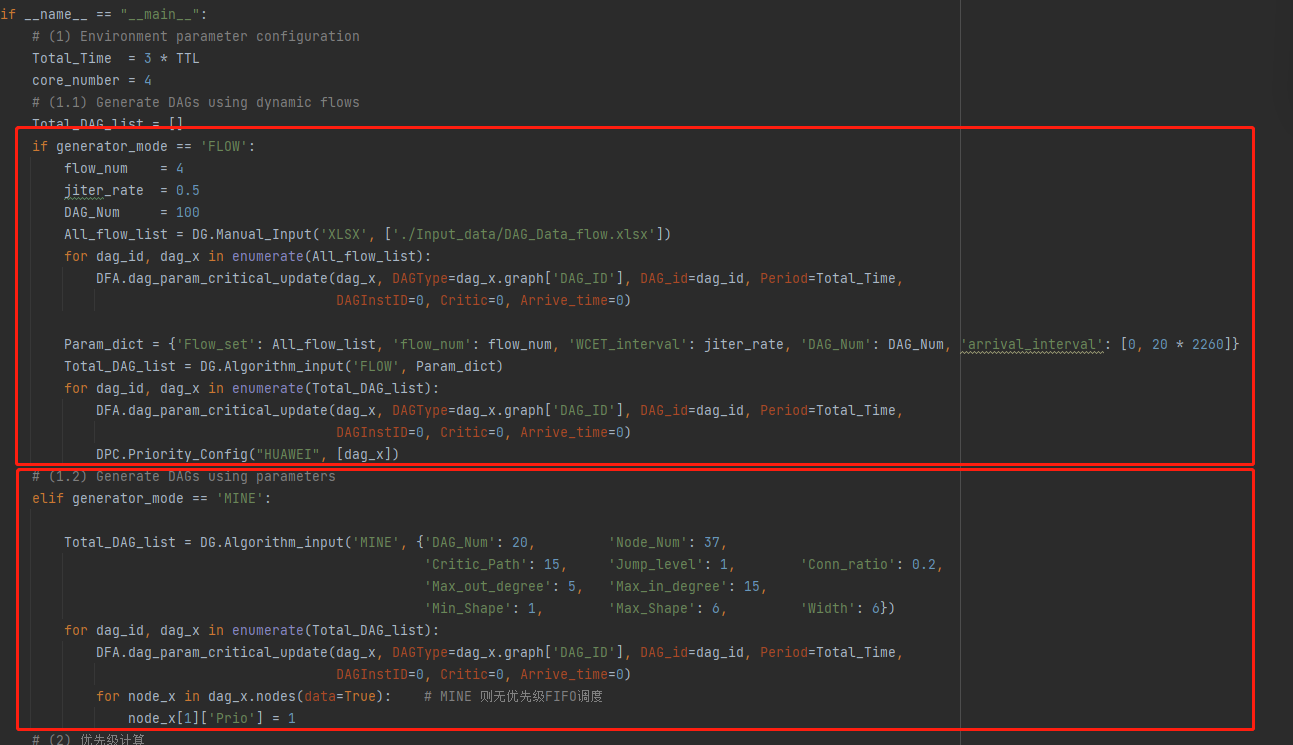
具体步骤入下

Step1. 点击DAG参数提取按钮，确定输入DAG特性数据的地址（输入数据的样例文件在.\Input\_data\Flow\_data中）；输入数据后，将自动生成DAG关键特性列表；

Step2. 在生成DAG数量编辑框中输入要生成的DAG数量、DAG中的流数以及结点的抖动幅度(0.0-1.0)，其中输入流集合数量是自动生成的，并点击打开按钮在输出\_Excal\_DAG\_Data\_文件路径中确定输出文件的地址；

Step3. 点击基于流生成参数生成DAG按钮（中间会有一段时间的延迟）。完成后输出成功对话框。

## DAG仿真器



#### Step 1 DAG生成阶段：

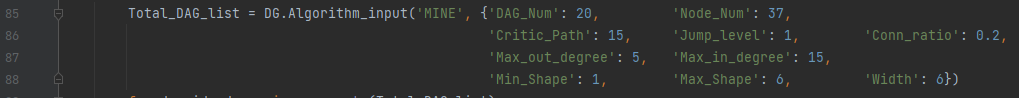
本阶段负责DAG生成，生成结果的DAG结合于：Total\_DAG\_list = []中。生成方式分为两种：

* 第一种的基于流的动态生成，该方法根据输入的流数据动态生成新的DAG,生成DAG的结点执行时间，DAG的层数，DAG的分支节点数量，DAG中各流的起始时间均会随机波动。相关参数配置如下图所示：



|  |  |
| --- | --- |
| 参数名称 | 参数简介 |
| Flow\_set | 输入的流集合，生成的DAG由这些流随机组合； |
| flow\_num | 生成DAG的流数量，是一个整数； |
| WCET\_interval | DAG各结点的执行时间上下抖动比例； |
| DAG\_Num | 总共生成的DAG的数量； |
| arrival\_interval | DAG中各流的到达时间随机分布区间，（是一个二位数组，为到达时间的随机分布的上下限）； |

第二种的基于参数生成DAG，该方法根据输入的关键参数利用生成器生成符合条件的DAG集合。相关参数配置如下图所示：



|  |  |
| --- | --- |
| 参数名称 | 参数简介 |
| DAG\_Num | 总共生成的DAG的数量； |
| Node\_Num | 生成DAG的结点数量； |
| Critic\_Path | 生成DAG的层数量； |
| Jump\_level | 生成DAG的结点间边的最大层跨度； |
| Conn\_ratio | 生成DAG的稠密度； |
| Max\_out\_degree | 生成DAG的最大出度； |
| Max\_in\_degree | 生成DAG的最大入度； |
| Min\_Shape | 生成DAG的最小shape； |
| Max\_Shape | 生成DAG的最大shape； |
| Width | 生成DAG的宽度； |

#### Step 2 DAG优先级计算阶段：

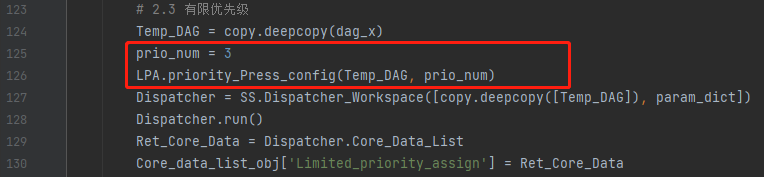
优先级计算函数如下所示：



可选的优先级分配模式如下表所示：

|  |  |
| --- | --- |
| 优先级算法 | 简介 |
| SELF | 提出基于结构感知的优先级算法； |
| WCET | 基于WCET的优先级分配算法，大结点优先； |
| HEFT | 基于HEFT的优先级分配算法，大权重结点优先； |
| He\_2019 | 基于He\_2019的优先级分配算法，长路径优先（有拓扑约束）； |
| He\_2021 | 基于He\_2021的优先级分配算法，长路径优先（无拓扑约束）； |

此外，在SELF的基础上，可以赋予有限优先级，接口如下：



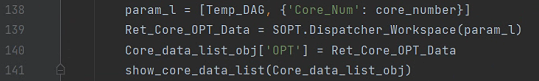
根据指定的优先级数量，分配有限优先级。

#### Step 3 DAG运行仿真阶段：

DAG分配好优先级后，则对DAG进行运行仿真，仿真参数param\_dict的参数列表如下所示：

|  |  |
| --- | --- |
| 参数名称 | 简介 |
| Core\_Num | 表示核心的数量 |
| Total\_Time | 表示仿真的总时间长，默认为DAG的周期长度； |
| Dynamic | 是否为动态调度（执行过程中检查是否重算优先级）； |
| Enqueue\_rank | 基于优先级的就绪队列入队排序，默认为False； |
| Priority\_rank | 基于优先级排序，默认为True； |
| PE | 预调度机制，默认为False； |
| Non\_WC | Non-work-conserving机制，默认为False； |

#### Step 4 DAG最优调度：



最优调度的参数param\_dict的参数列表如下所示：

|  |  |
| --- | --- |
| 参数名称 | 简介 |
| Temp\_DAG | 表示仿真的DAG对象 |
| Core\_Num | 表示核心的数量 |

通过函数SOPT.Dispatcher\_Workspace(param\_l)生成最优调度各核心执行时间的数据列表：Core\_data\_list\_obj

#### Step 4 DAG运行仿真结果：



将所有仿真的各核心执行时间数据列表输入，输出执行的根特图，分别列出不同调度的结果：

