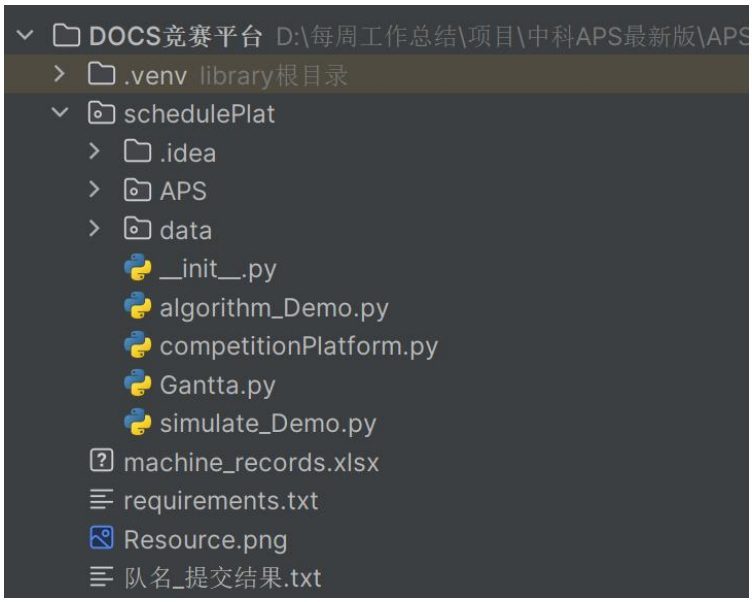
动态调度竞赛仿真平台

(一) 环境准备

仿真平台性能	处理器：i9-13900H　2.60 GHz 内存：16GB RAM 存储：1GB 可用磁盘空间 显卡：　(无专用显卡)
软件环境	操作系统：Windows 10/11 (64 位) Python 版本：Python 3.12 (必须) 推荐 IDE: VS Code (轻量级) PyCharm Professional (高级功能)
必要算法库	contourpy==1.3.2 cycler==0.12.1 et_xmlfile==2.0.0 fonttools==4.58.4 intervaltree==3.1.0 kiwisolver==1.4.8 matplotlib==3.10.3 numpy==2.3.0 openpyxl==3.1.5 packaging==25.0 pandas==2.3.0 pillow==11.2.1 pyparsing==3.2.3 python-dateutil==2.9.0.post0 pytz==2025.2 six==1.17.0 sortedcontainers==2.4.0 tzdata==2025.2

(二) 项目目录介绍




```
schedulePlat/
├──── .venv/                  # Python 虚拟环境目录
├──── .idea/                  # IDE 配置文件 (PyCharm 等)
├──── APS/                    # 仿真平台核心组件
├──── data/                   # 竞赛实例数据存储
|
├──── algorithm_Demo.py       # 参赛选手算法文件模板
├──── competitionPlatform.py  # 仿真平台主程序
├──── Gantta.py               # 甘特图绘制库
├──── simulate_Demo.py        # 竞赛仿真主程序
├──── __init__.py             # Python 包初始化文件
|
├──── machine_records.xlsx    # 仿真过程记录数据
├──── requirements.txt        # 项目依赖库清单
├──── Resource.png            # 机器甘特图资源示例
└──── 队名_提交结果.txt        #初赛仿真结果
```

(三) 基础使用说明

该程序用于参赛队伍，对参赛算法进行调试，确保提交代码正确。

1. 配置竞赛案例

```python
# 0. 配置竞赛案例
instance_name = 'num1000_lam0.03_change0__7.txt'
instance_names= []

path = os.path.join('schedulePlat', 'data', 'instance', 'competition', instance_name)

# 打印当前工作目录和文件路径，以便调试
print("当前工作目录是:", os.getcwd())
print("尝试打开的文件路径是:", path)
```

配置要仿真的实例的路径

2. 创建仿真平台实例

```python
# 1. 创建仿真平台实例
platform = CompetitionPlatform()
```

3. 创建参赛队伍算法实例

```python
# 2. 创建参赛队伍算法实例
from algorithm_Demo import SchedulingAlgorithm
team_algorithm = SchedulingAlgorithm()
```

algorithm_Demo 为主办方提供的参赛算法例子，参赛队伍需要自行设计算法

4. 运行仿真

```python
# 4. 运行仿真
result = platform.run_simulation(path, team_algorithm, isTimeout: False)
```

5. 输出仿真结果

```python
# 5. 输出结果
orders = platform.getOrders()
print("\n仿真结果:")
print(f"订单达成率: {orders['fulfillment_rate'].values[0]:.2%}")
platform.getGantta( startTime: 600, endTime: 900)
machine_records = platform.getMachineRecord()
with pd.ExcelWriter('machine_records.xlsx') as writer:
    for sheet_name, df in machine_records.items():
        df.to_excel(writer, sheet_name=str(sheet_name), index=False)
```

6. 初赛仿真结果

```python
    # 6 记录最终结果数据，保存在txt。
    print(f"{instance_name},{orders['fulfillment_rate'].values[0]},{result}", )
```

请将每个 instance 的输出结果，保存在“队名_提交结果.txt”中，将用于初赛的最终评审。

(四) 竞赛仿真平台接口说明

1	def run_simulation(self, path, algorithm_module, isTimeout=True) -> dict:
	运行仿真案例

	:param path: 仿真案例的路径 :param algorithm_module: 动态调度算法 :param isTimeout: 是否开启 30s 实时调度限制 :return: 返回仿真结果（数字签名）
2	def getGantta(self, startTime, endTime)
	生成甘特图 :param startTime: 开始时间点 :param endTime: 结束时间点
3	getMachineRecord(self) -> Dict[str, pd.DataFrame]
	获取所有机器的历史作业记录。 该方法返回一个字典，包含每台机器已分配的所有任务信息。 Returns: Dict[str, pd.DataFrame]: 以机器 ID 为键的字典，值为包含该机器所有作业记录的 DataFrame， DataFrame 包含以下列： - task_id: 任务 ID - start_time: 任务开始时间 - end_time: 任务结束时间
4	getOrders(self, only_unfinished=True) -> pd.DataFrame
	获取订单数据及相关状态信息。 该方法返回包含订单信息的 DataFrame，包含订单基本信息、当前进度状态以及全系统订单完成率。 Args: only_unfinished (bool, optional): 是否只返回未完成的订单。默认为 True。 - True: 仅返回尚未完成的订单 - False: 返回所有订单（包括已完成的） Returns: pd.DataFrame: 包含订单信息的 DataFrame，包含以下列： - order_id: 订单唯一标识 - product_type: 产品类型 - arrival_time: 订单到达时间 - due_date: 订单交期 - current_stage: 当前处理工序（如果当前没有进行中的工序则为上一个工序的结果。若首工序也未开始，则为 None） - assigned_machine: 分配到的机器 ID（如果当前没有分配则为为上一个工序的分配结果。若首工序也未开始，则为 None） - start_time: 当前工序开始时间（如果未开始则为上一个工序的开始时间。若首工序也未开始，则为 None） - end_time: 当前工序结束时间（如果未开始则为上一个工序的结束时间。若首工序也未开始，则为 None） - fulfillment_rate: 当前系统订单达成率（已完成订单/所有已到达订单）
5	getCurrentMachineStatus(self) -> pd.DataFrame

	获取当前时刻所有机器的状态信息。 该方法返回一个 DataFrame，描述在当前时间点各机器的状态（空闲或正在执行的任务信息）。 Returns: pd.DataFrame: 包含每台机器当前状态的 DataFrame，包含以下列: - task_id: 当前执行的任务 ID（如果机器空闲则为 None） - start_time: 当前任务的开始时间（如果机器空闲则为 None） - end_time: 当前任务的结束时间（如果机器空闲则为 None）
6	getMBOM(self) -> pd.DataFrame:
	获取制造 BOM(Bill of Materials)信息。 该方法从仿真实例中提取产品的工艺路线信息，包括每个产品在各生产阶段可选用的设备和相应处理时间。 Returns: pd.DataFrame: 包含制造 BOM 信息的 DataFrame，包含以下列: - product_type: 产品类型标识（格式为"0"、"1"、"2"等） - stage: 生产阶段标识（格式为"0"、"1"、"2"等） - machine_id: 可用于该工序的设备 ID - process_time(s): 在该设备上完成该工序所需的处理时间（秒）
7	getSimulationTime(self) -> float:
	获取当前仿真时间（从仿真开始起经过的时间）。 该方法计算从仿真基础时间点（通常是仿真启动时间）到当前时刻经过的时间。 Returns: float: 仿真经过的时间（以秒为单位）

(五) 竞赛算法定义

参赛算法需要定义为 python 的类，必须包含如下函数，可以参考示例 algorithm_Demo.py:

def generate_schedule(self, platform) -> pd.DataFrame
动态生成调度计划 :param platform: 仿真测试平台 :return: 调度计划 DataFrame schedule_df: 调度计划 DataFrame，包含以下列: - task_id: 任务 ID (格式: order_id-process_id) - machine_id: 设备 ID - start_time: 计划开始时间

(六) 注意事项

A. 开发建议:

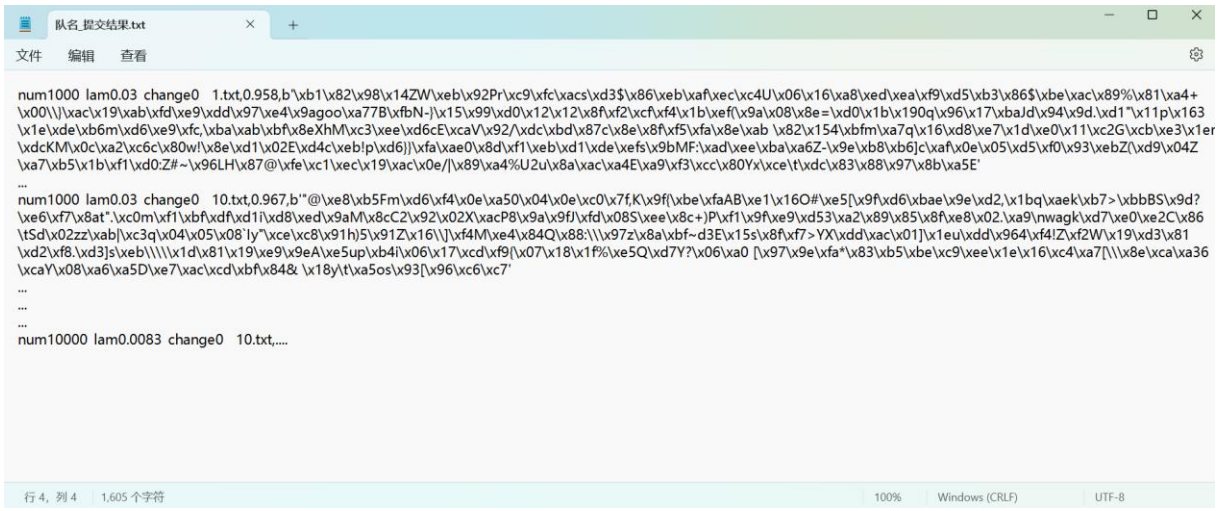
- 在 algorithm_Demo.py 中实现 SchedulingAlgorithm 类

- 保留框架原有接口方法
- 开发过程中设置 isTimeout=False 避免调度限制

B. 提交前检查:
- 确保 isTimeout=True 时算法能在 30 秒内完成调度
- 验证结果是否包含必要字段
- 清理调试输出语句

(七) 提交规范

A. 必交内容
- 主算法文件（队名_Algorithm.py）：必须包含完整的 SchedulingAlgorithm 类实现
- 说明文档（队名_doc.pdf）：简述算法思路和关键函数
- 依赖文件(requirements.txt): 列出非系统级 Python 库
- 初赛仿真结果（队名_提交结果.txt）：由竞赛仿真平台输出的结果，用于初赛评审。
  - 应包含所有测试实例的结果，每行对应一个实例，格式如下：
    [实例名称],[订单达成率],[数字签名]
    实例名称: 主办方提供的 instance 文件名（如 num1000_lam0.03_change0_1.txt）
    订单达成率：该实例仿真的最终结果，保留小数点后三位（如 0.958）
    数字签名：由仿真程序生成的验证字符串（不可修改）
  - 完整示例：



  - 关键要求：
    必须包含所有测试实例的结果（缺一不可）
    实例顺序不限，但每行对应一个完整实例
    不可修改签名内容（任何修改都将导致验证失败）
    数字签名需保持原始格式（含开头的 b'和结尾的'）

B. 提交方式：
- 邮箱提交：
  - mail: wangrui@sia.cn
  - 提交压缩包命名：团队名_提交日期.zip