**环境基本框架**

类 PowerGridEnv:

初始化(电网数据, 高风险路径数量s):

加载电网拓扑数据

初始化OPF求解器

初始化状态变量

设置高风险路径数量s

重置():

恢复电网到初始状态

清空已断开边的记录

当前动作计数器归零

返回初始观测值

执行动作(动作):

边索引 = 动作

断开指定边

模拟级联故障过程

计算奖励和新状态

记录当前路径

检查是否完成一条完整路径

返回新观测值、奖励、是否结束、额外信息

模拟级联故障():

实现Algorithm 1的级联故障模拟逻辑

返回总负荷损失

**状态空间设计**

获取观测值():

电网当前状态表示 = {

"拓扑特征": 包含节点和边的特征矩阵,

"节点特征": [电压, 负荷, 发电量],

"边特征": [当前潮流, 阻抗],

"已断开边": 已选择断开的边集合,

"当前步骤": 当前是第几个动作(0-2之间)

}

if 使用GAT:

返回GAT可处理的图结构数据

else:

返回扁平化状态向量

**动作空间设计**

定义动作空间():

可选动作 = 电网中所有未断开的边

return 可选动作

动作掩码():

掩码 = np.ones(边总数, dtype=bool)

for 边 in 已断开的边:

掩码[边索引] = False

return 掩码

**奖励设计**

计算奖励(当前负荷损失, 上一步负荷损失, 当前步骤):

单步奖励 = 当前负荷损失 - 上一步负荷损失

if 当前步骤 == 2: # 最后一步

终局奖励 = 当前负荷损失的额外倍数 # 鼓励找到总体高损失路径

return 单步奖励 + 终局奖励

else:

return 单步奖励

**级联故障模拟实现**

处理级联故障(初始断开边):

# 1. 初始断开指定边

断开边(初始断开边)

# 2. 记录初始状态

总负荷损失 = 0

迭代次数 = 0

最大迭代次数 = 50 # 防止无限循环

初始孤岛列表 = 检测孤岛(断开边后的电网)

# 3. 级联故障主循环

while 迭代次数 < 最大迭代次数:

迭代次数 += 1

系统变化 = False

# 3.1 检测当前孤岛

if 迭代次数 == 1

当前孤岛列表 = 初始孤岛列表

else

当前孤岛列表 = 新的孤岛列表

新的孤岛列表 = []

# 3.2 处理每个孤岛

for 孤岛 in 当前孤岛列表:

# 执行供需平衡计算

负荷削减变化 = 执行负荷削减(孤岛)

总负荷损失 += 负荷削减变化

# 计算潮流

计算AC潮流(孤岛)

# 频率保护

因频率变化而造成的负荷削减 = 处理频率问题(孤岛)

总负荷损失 += 因频率变化而造成的负荷削减

# 电压保护

因电压变化而造成的负荷削减 = 处理电压问题(孤岛)

总负荷损失 += 因电压变化而造成的负荷削减

if 负荷削减变化 or 因频率变化而造成的负荷削减 or 因电压变化而造成的负荷削减:

系统变化 = True

# 3.3 检查线路过载

过载支路列表 = 检查支路过载(孤岛)

# 3.4 如果有过载支路，断开最严重的过载支路

if 过载支路列表:

新的孤岛 = 断开支路(孤岛，过载支路列表)

新的孤岛列表.append(新的孤岛)

系统变化 = True

# 如果没有过载支路且系统无变化，则级联过程结束

if not 过载支路列表 and not 系统变化:

break

# 4. 计算最终负荷损失

最终总负荷 = 计算系统总负荷()

总负荷损失 = 初始总负荷 - 最终总负荷

return {

"负荷损失": 总负荷损失,

"迭代次数": 迭代次数,

"最终孤岛数": len(检测孤岛()),

"断开的边": 获取所有断开的边()

}

**孤岛检测和处理**

检测孤岛(断开边后的电网):

使用深度优先搜索或广度优先搜索算法

识别电网中所有连通子图

return 孤岛列表

处理孤岛(孤岛):

# 孤岛中负荷与发电是否平衡

发电能力 = 计算孤岛中总发电能力()

负荷需求 = 计算孤岛中总负荷需求()

if 发电能力 < 负荷需求:

# 需要负荷削减

削减比例 = 发电能力 / 负荷需求

for 负荷节点 in 孤岛:

调整负荷(节点, 削减比例)

return 削减后的负荷损失

**低频减载(UFLS)和低压减载(UVLS)**

执行低频减载(孤岛):

频率偏差 = 计算频率偏差(孤岛)

if 频率偏差 > 0.2 Hz:

减载比例 = 计算基于频率偏差的减载比例()

for 负荷节点 in 孤岛:

当前负荷 = 获取节点负荷(节点)

减载量 = 当前负荷 \* 减载比例

调整负荷(节点, 当前负荷 - 减载量)

return 减载后的负荷损失

执行低压减载(孤岛):

for 母线 in 孤岛:

if 母线电压 < 0.9标幺值:

减载比例 = 计算基于电压偏差的减载比例()

当前负荷 = 获取节点负荷(母线)

减载量 = 当前负荷 \* 减载比例

调整负荷(母线, 当前负荷 - 减载量)

return 减载后的负荷损失

**支路过载处理**

检查支路过载(电网（结构体）):

过载支路列表 = []

得到‘所有支路’和‘支路潮流’

for 支路 in 所有支路:

if 支路潮流 > 支路容量限制（一个固定的类变量）:

过载支路列表.append(支路)

return 过载支路列表

断开支路(电网（结构体），过载支路列表):

if 过载支路列表 非空:

for 支路 in 过载支路列表

设置电网（结构体）中表达支路连通性的数据的相应位置为False(支路)

return 断开支路的电网

return ‘过载支路列表为空’