航司一体化调度：MA-HRL 架构+拓扑-GRPO 算法完整方案

一、航司一体化调度 MA-HRL 完整架构

本架构采用分层式多智能体协同+全局协调的设计思路，贴合航司调度层级决策逻辑，兼顾安全硬约束与全局优化目标，架构分为四层，层级间数据单向流转、同级智能体分布式通信。

（一）整体架构拓扑

（二）各层级核心职能

1. 全局协调智能体

唯一顶层决策单元，不参与底层执行，负责输出全局优化目标、跨智能体资源配额、冲突协调规则，平衡准点率、运营成本、旅客体验三大核心目标，向下层智能体下发子任务目标。

2. 中层多智能体集群

四类平行智能体独立决策、局部通信，各司其职且相互联动：

- 航班调度智能体：负责航班时刻调整、机型匹配、航班串优化
- 机队调度智能体：负责飞机轮转、维修窗口插入、机队资源调配
- 机组调度智能体：负责机组配对、执勤时长管控、休息合规调度
- 航班恢复智能体：负责极端天气/流控/故障下的应急重排、次生延误管控

3. 硬约束执行过滤层

刚性拦截所有违规动作，采用动作掩码+惩罚机制，杜绝违反民航法规、飞行时长、维修适航、机场时刻窗等硬约束行为，保障决策可落地、合规性。

4. 环境与数据感知层

整合航班运行数据、机队状态、机组信息、天气流控、机场拥堵等多源数据，完成特征提取与降维嵌入，为所有智能体提供标准化状态输入。

二、拓扑-GRPO 航司调度拓扑图定义

基于 MA-HRL 架构，将智能体调度关系转化为动态有向拓扑图，解决多智能体通信冗余、状态空间爆炸问题，实现稀疏化协同。

（一）拓扑图形式化定义

定义航司调度拓扑图为 G = (V, E, S)，其中：

1. 节点集 V：对应中层多智能体，V = {V_f, V_p, V_c, V_r}

- V_f：航班调度智能体节点
- V_p：机队调度智能体节点
- V_c：机组调度智能体节点
- V_r：航班恢复智能体节点

2. 边集 E：智能体间通信耦合关系，E_ij ∈ {0,1}，E_ij = 1 表示智能体 i 向智能体 j 传输状态信息，E_ij = 0 表示无通信；通过策略网络动态学习边的生成，保留强耦合关联、剔除冗余连接。

3. 全局状态 S：拓扑图输入特征，包含航班计划、机队位置、机组执勤时长、天气流控、机场时刻、旅客中转信息六类核心状态。

（二）拓扑生成规则

1. 采用图注意力网络（GAT）生成节点嵌入，计算智能体间耦合权重
2. 设定稀疏阈值，仅保留权重前 K 的通信边，避免全连接拓扑
3. 应急场景下自动扩增局部边，正常场景保持极简拓扑，自适应调度场景变化

三、拓扑-GRPO 航司调度核心伪代码

采用无 Critic、分组相对优势优化，适配分层多智能体拓扑，兼顾训练稳定性与计算效率，可直接用于代码开发。

```
# 拓扑-GRPO 航司一体化调度核心算法伪代码

import torch

import numpy as np


# 初始化参数

MAX_EPISODES = 10000  # 最大训练轮次

K = 8  # 每组采样拓扑数量

LEARNING_RATE = 3e-4
```

```
KL_COEFF = 0.1  # KL 散度约束系数
EPS = 0.2  # PPO 裁剪系数

# 初始化网络：拓扑策略网络+动作策略网络
policy_net = GraphPolicyNetwork()  # 输出动态通信拓扑+智能体动作
old_policy_net = GraphPolicyNetwork()
old_policy_net.load_state_dict(policy_net.state_dict())
optimizer = torch.optim.Adam(policy_net.parameters(), lr=LEARNING_RATE)

# 主训练循环
for episode in range(MAX_EPISODES):
    # 1. 从环境感知层获取全局状态 S
    global_state = env.get_state()
    topology_list = []
    action_list = []
    reward_list = []

    # 2. 同一状态下采样 K 组不同拓扑+动作
    for k in range(K):
        # 生成动态通信拓扑
        topology = policy_net.sample_topology(global_state)
        # 基于拓扑生成各智能体动作（经硬约束过滤）
        action = policy_net.sample_action(global_state, topology, hard_constraint_mask)
        # 执行动作，获取全局奖励
        reward = env.step(action, topology)
```

```
        topology_list.append(topology)
        action_list.append(action)
        reward_list.append(reward)

    # 3. GRPO 核心：组内相对优势函数计算
    reward_mean = np.mean(reward_list)
    reward_std = np.std(reward_list) + 1e-8
    advantage = [(r - reward_mean) / reward_std for r in reward_list]

    # 4. 计算拓扑-GRPO 损失函数
    loss_total = 0
    for k in range(K):
        # 策略概率比值
        ratio = policy_net.get_prob(topology_list[k], action_list[k],
global_state) / \
                old_policy_net.get_prob(topology_list[k], action_list[k],
global_state)
        # PPO 裁剪损失
        clip_loss = torch.min(ratio * advantage[k], torch.clamp(ratio, 1-EPS,
1+EPS) * advantage[k])
        # KL 散度约束（防止拓扑突变）
        kl_loss = policy_net.kl_divergence(old_policy_net, global_state)
        # 总损失
        loss_total += -torch.mean(clip_loss) + KL_COEFF * kl_loss

    # 5. 策略网络更新
    optimizer.zero_grad()
    loss_total.backward()
    optimizer.step()
```

```
    old_policy_net.load_state_dict(policy_net.state_dict())


    # 6. 模型保存与日志输出
    if episode % 100 == 0:
        torch.save(policy_net.state_dict(),
f'topology_grpo_airline_{episode}.pth')
        print(f"Episode:{episode}, Loss:{loss_total.item()},
Avg_Reward:{reward_mean}")
```

四、航司调度完整奖励函数设计

采用多目标加权奖励，兼顾全局优化与刚性约束，区分正向激励与负向惩罚，贴合航司实际运营指标。

（一）总奖励公式

$$R_{total} = \omega_1 R_{on\text{-}time} + \omega_2 R_{cost} + \omega_3 R_{crew} + \omega_4 R_{recovery} - R_{penalty}$$

其中 $\omega_1,\omega_2,\omega_3,\omega_4$ 为加权系数，满足 $\omega_1 + \omega_2 + \omega_3 + \omega_4 = 1$，可根据运营侧重调整。

（二）子奖励函数定义

1. 准点率奖励 $R_{on\text{-}time}$

$$R_{on\text{-}time} = \frac{N_{on\text{-}time}}{N_{total}}$$

$N_{on\text{-}time}$ 为准点航班数量， N_{total} 为总航班数，准点率越高奖励越大。

2. 运营成本奖励 R_{cost}

$$R_{cost} = 1 - \frac{C_{actual}}{C_{base}}$$

C_{actual} 为实际运营成本（燃油+机组+延误成本）， C_{base} 为基准成本，成本越低奖励越高。

3. 机组合规奖励 R_{crew}

$$R_{crew} = \frac{N_{crew\text{-}compliance}}{N_{crew\text{-}total}}$$

衡量机组执勤时长、休息间隔合规率，合规度越高奖励越高。

4. 航班恢复奖励 $R_{recovery}$

$$R_{recovery} = 1 - \frac{T_{delay}}{T_{max\text{-}delay}}$$

针对延误场景，衡量延误时长管控效果，延误越短奖励越高。

5. 违规惩罚项 $R_{penalty}$

触发硬约束时施加大额惩罚：违反飞行时长合规罚 100、违反维修适航罚 200、超出机场时刻窗罚 150，无违规则惩罚为 0。

五、系统实现与部署方案

（一）技术栈选型

1. 核心框架

- Python 3.8+
- PyTorch 1.10+
- NumPy 1.20+
- Pandas 1.3+

2. 部署环境

- 云端：Kubernetes 集群 + Docker 容器化部署
- 边缘端：轻量化模型推理服务

（二）性能优化策略

1. 分布式训练

- 采用 Horovod 进行多 GPU 并行训练
- 数据并行 + 模型并行混合策略

2. 在线学习机制

- 增量学习更新模型参数
- 经验回放池存储历史决策样本

（三）安全与合规保障

1. 决策审计

- 完整记录所有调度决策过程
- 建立决策追溯机制

2. 安全防护

- 输入数据异常检测
- 决策结果合规性验证

六、预期效果与评估指标

（一）核心性能指标

1. 调度效率提升

- 决策响应时间：≤ 2 秒/次
- 调度方案生成成功率：≥ 99.5%

2. 运营指标改善

- 准点率提升：≥ 8%
- 运营成本降低：≥ 12%
- 机组合规率：≥ 99.9%

（二）系统稳定性指标

1. 可靠性

- 系统可用性：≥ 99.99%
- 故障恢复时间：≤ 30 秒

2. 扩展性

- 支持最大航班量：≥ 5000 架次/天
- 支持最大机队规模：≥ 500 架飞机

七、实施路线图

阶段一：原型验证（1-3 个月）

- 完成核心算法开发
- 构建仿真测试环境
- 验证基础功能

阶段二：系统集成（4-6 个月）

- 与现有系统对接
- 完成安全合规测试
- 建立运维监控体系

阶段三：全面推广（7-12 个月）

- 全航司范围部署
- 建立持续优化机制
- 形成标准化解决方案

文档版本： V1.0

编制日期： 2026 年

适用范围： 航空公司一体化调度系统设计与实施