Think Global, Act Local - Agent-Based Inline Recovery for Airline Operations

Yashovardhan S. Chati
Tata Consultancy Services Research
Pune, India
ys.chati@tcs.com

Ramasubramanian Suriyanarayanan
Tata Consultancy Services Research
Chennai, India
ramasubramanian.suriyanarayanan@tcs.com

Arunchandar Vasan
Tata Consultancy Services Research
Chennai, India
arun.vasan@tcs.com

ABSTRACT

Flight delays can significantly affect airline operations. Airlines use inline recovery actions (e.g., speeding up aircraft cleaning) to mitigate the effect of flight delays. Inline (or tactical) recovery for disruptions mostly relies on human expertise that is locally optimal (e.g., at an airport-level). Because an airline is a complex and stochastic network of dependencies, a locally optimal action may be globally sub-optimal. Considering global effects for inline recovery is computationally challenging with conventional algorithmic approaches for optimization. We complement existing approaches with Stochastic Minplus with State (SMS), a novel agent-based approach for inline recovery. SMS generalizes message passing algorithms for a state-dependent stochastic airline network with resource constraints. We evaluate our approach on a real-world airline network with around 4,000 flights per day for two regimes with 1) normal delays, and 2) interrupted operations. As baselines, we use approaches based on greedy local optimality, integer programming (IP), and constraint programming (CP). Our evaluation shows that: 1) a globally informed SMS improves over greedy locally optimal approach by 24.7% in quality, 2) SMS achieves solutions that are better by 14.1% (10.6%) in quality and 7x (9x) in computation time over IP (CP) with timeout, and 3) SMS achieves solutions within 5% of the optimal solution for simpler problem instances.

KEYWORDS

Airline operations; Irregular operations recovery; Inline optimization; Message passing algorithm

ACM Reference Format:
Yashovardhan S. Chati, Ramasubramanian Suriyanarayanan, and Arunchandar Vasan. 2024. Think Global, Act Local - Agent-Based Inline Recovery for Airline Operations. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Controllable delays: Flight delays are estimated to have cost airlines and passengers (PAX) in the USA around $33 billion in 2019 [2]. Airlines need to pay for perishable airport usage time. Further, PAX and crew inherit delays of their flights, and may miss connections at hub airports. At a planning level, an airline schedule aims to handle delays through in-built buffers in the schedule. However, despite the best of planning, flight operations deviate from the plan. For example, in 2019, about 21% flights in the USA had more than 15 minutes of arrival delay [26]. At an operational level, delays occur *despite* the schedule buffers. About 70% of operational delays are estimated to be under an airline's control [1].

Inline recovery: When an incoming flight arrives late, an airline has a few *inline control levers* to handle the delay. These include deploying additional resources for cleaning the aircraft, holding a departing flight for connecting PAX, etc. (details in Section 2). These control levers come at a cost (e.g., of labor) and an associated benefit (e.g., reduced departure delay). The *inline/tactical recovery* problem, therefore, is to dynamically decide the appropriate recovery action for a flight at the Airline Operations Control Center. Because the airline network is complex and individual actors (e.g, a gate manager) have individual airport-level local metrics for optimality (e.g, turn-around delay for gate operations at that airport), the decision making for inline recovery in practice is highly localized at the airport-level, typically manual, and based on rules of thumb and department-specific performance targets [12].

Global vs local: While local and manual decision making for inline recovery is fast, it is most likely sub-optimal. First, the decision maker cannot take the perspective of the entire airline network due to the sheer complexity involved. Second, even if they intuitively understand the current state of the network (e.g., congestion by time-of-day patterns), the effect of the decisions may appear later in the network and not be directly quantifiable manually. Third, different human operators may make varying and highly subjective calls on local inline recovery for the same network state leading to unpredictable operations. As we show in our evaluation, a locally optimal action may not be globally efficient. Therefore, there is a need for automated decision support for global inline recovery.

Challenges: Automated global inline recovery is non-trivial due to the complex dependencies between local actions and global effects. First, the reward for actions is non-linear. For example, delayed connecting PAX either miss or don't miss the connection depending on the time for which a departing flight is held. Second, the action taken for recovering one flight affects other flights locally. For example, a flight spending more time at a gate than scheduled would delay a subsequent flight at the same gate. Third, the action taken for recovering one flight affects other flights globally. A flight f_j departing late (primary delay) from an airport A_j to A_i could result in a subsequent late departure (secondary delay) of the next flight f_i departing from A_i. Fourth, airport resources for recovery

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

are constrained. Finally, incoming airline crew need to be eligible for operating their outgoing next flight. Excess delay might push them off a shift boundary and make them illegal for a take-off. Thus, local actions cause higher-order global effects and considering them for decision making can be computationally challenging.

Problem statement: At time t, given the currently known global state s_t of a set of flights as a boundary condition, our goal is to identify, in a computationally efficient manner, the global inline recovery actions of all flights of the airline, departing across all airports, over a future optimization horizon $[t{+}L, t{+}L{+}H]$. In practice, the problem would be solved by the operations control center of the airline using data feeds from all airports. This computation is to be repeated continuously in a rolling horizon fashion for $t = 0, \delta, 2\delta, \cdots$. To handle the global effect, we choose a suitably large horizon H as described in Section 6. To handle randomness, the stochastic optimization objective is to minimize a weighted combination of *expected* business performance metrics (departure delays of flights and missed PAX connections), and the *deterministic* cost of recovery actions. The constraints need to handle delay propagation and resource availability for recovery at airports. Because resources for inline recovery need to be moved across an airport, decision making and implementation need to be separated by a lead-time L. Because the operating conditions are highly dynamic, the control timestep δ cannot be too large. The time for decision making is limited, thus making conventional optimization unsatisfactory either in the compute time required or the quality of solution obtained. Thus, heuristic methods are resorted to but they may not always incorporate the uncertainty in airline operations.

Our approach: We overcome these challenges and complement existing works with SMS (Stochastic MinPlus with State), a scalable message-passing algorithm for inline recovery, based on the MaxPlus algorithm [27]. Our focus in this work is on reducing aircraft delays and missed PAX connections. In MaxPlus, agents $\mathcal{A}_i$ coordinate their actions to maximize a joint objective with purely action-dependent cost functions defined on nodes and edges of the coordination graph. In SMS, we generalize MaxPlus for airline operations in the following ways. First, we exploit the structure of the problem explicitly using a coordination graph for airline operations to capture the interactions between flights and PAX. Second, we augment MaxPlus with a state (the *inherited delay* of an aircraft, explained in Section 2). Third, we incorporate uncertainty by evaluating the message passing payoff functions as stochastic averages of various optimization scenarios in a simulation environment to identify the common optimal actions that *minimize* (hence, the *Min*Plus) the expected *costs* across all the scenarios. Last, we model resource constraints for inline recovery through a novel way of resolving potential conflicts using the payoff functions.

Implementation and evaluation: We evaluate our approach on a simulated environment calibrated with real-world data from a leading North American airline (anonymized for business requirements) with about 4,000 flights a day across 126 airports. We consider the following approaches as baselines for SMS: a do-nothing NOOP approach that does no inline recovery, a Greedy approach that is locally optimal, and two conventional optimization approaches based on constraint programming (CP) and integer (non-linear) programming (IP). All the algorithms except NOOP are implemented in a model-predictive rolling horizon fashion with (forecasted) scenario-based optimization. Each approach is finally evaluated over a set of stochastic evaluation scenarios that is different from the set of forecasted scenarios but the same across the tested algorithms. The performance metrics are: 1) a combination of business metrics (cost of departure delays, cost of missed PAX connections, and cost of recovery actions), and 2) the computation time. To summarize, our **specific contributions** include the following:

- We model inline recovery for intelligent airline operations as a stochastic optimization problem that captures higher-order network-wide effects of airport-level local recovery actions.
- We design an agent-based approach for integrated inline recovery in airline operations. Our approach exploits domain knowledge encoded as a coordination graph to achieve scale for real-time decision making.
- We evaluate our approach on a simulated environment calibrated with real-world airline data against several baselines for two different operating regimes: normal delays due to business as usual (BAU) and higher delays due to irregular operations (IROP) at geographically close airports.

Our **key findings** include the following:

- A globally informed SMS outperforms a locally optimal Greedy approach by 26.3% (24.7%) in the BAU (IROP) regime.
- On realistic problem instances where CP and IP need a timeout, in the IROP regime, SMS improves over IP (CP) by 14.1% (10.6%) in business cost, and by 7x (9x) in compute time. On simpler problem instances where CP or IP completes to optimality, SMS is within 5% of the optimal CP or IP cost.
- All globally informed control strategies (SMS, CP, IP) identify a sweet-spot that occurs due to the domain constraints as the result of two opposing network effects: increasing global delays and reducing global impact of delays.
- SMS achieves scale because 1) its complexity for the airlines domain can be approximated by a linear function of the problem size (in terms of the number of flights in the coordination graph), and 2) it handles resource constraints in a novel way through the message passing functions. CP and IP fail to scale because their constraints and variables grow non-linearly with the problem size.

The rest of the paper is organized as follows. Section 2 presents a brief background of integrated inline recovery in airline operations and surveys related work. Section 3 presents the optimization problem statement formally. Section 4 presents the details of our solution. Section 5 presents the experimental setup used for evaluation. Section 6 presents the results of the evaluation and discusses the limitations of our approach. Section 7 concludes.

2 BACKGROUND

Actions for inline recovery: Airline disruption recovery is a well-researched problem in literature, as reviewed in [14]. Such a disruption recovery involves schedule recovery, aircraft recovery, crew recovery, passenger recovery, or a combination of two or more. Studies have used a variety of recovery actions such as flight delay, flight cancellation, aircraft swap, cruise speed control, PAX itinerary change, and quick ground turn-around [14]. Quick ground turn-around has shown promise in reducing flight delays

[24] and intentional delay (i.e., 'hold') of connecting flights can reduce missed PAX connections [20, 24]. A recovery action has a cost (typically, airport resources or personnel) and a specific impact on the delay (increase/decrease). Common actions used for inline recovery that we consider in this paper are:

- **Surge**: Additional cleaning crew, baggage handlers, etc. speed up the turn-around process. Due to resource constraints, only few flights can surge simultaneously at an airport.
- **Parallelize**: The turn-around process can be sped up by allowing PAX to board when the aircraft is being refueled. Because this needs the fire brigade to be near the aircraft, only few flights can be parallelized simultaneously.
- **Hold**: The departing flight waits at a gate for incoming PAX. The resource cost is the gate time. If the gate is shared across airlines, the airport can impose a delay cost on the waiting airline. Hold can be of various durations (we choose 5 min, 10 min, 15 min) and each is a separate action.
- **No intervention (NOOP)** is always a feasible action. Section 5 describes the delay change and resource cost of each of the six possible actions in our model.

Effect of recovery actions: In airline operations, an incoming flight is "turned around" as an outgoing flight after deboarding, cleaning, baggage handling, boarding, etc. Figure 1 illustrates turn-around processes of two aircraft at an airport. A tail or physical

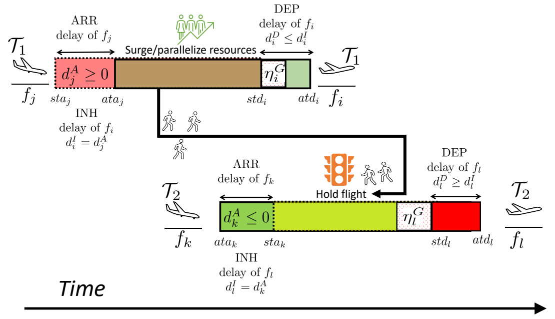


Figure 1: Turn-around process and inline recovery

aircraft $\mathcal{T}_1$ arrives as flight f_j at the actual (scheduled) time of arrival ata_j (sta_j), and takes off as flight f_i at the actual (scheduled) departure time atd_i (std_i). The arrival delay $d_j^A \triangleq ata_j - sta_j$ is the difference between actual and scheduled times of arrival of f_j. The arrival delay d_j^A of flight f_j becomes the *inherited delay* d_i^I of flight f_i. If $d_i^I \geq 0$ and the schedule between f_j and f_i has no buffer time, then the departure delay is given by $d_i^D = d_i^I + \eta_i^G$, where η_i^G is a random ground delay (in this case, shown as positive). However, if we speed up the turn-around process by σ_i by surging/parallelizing resources as an inline recovery, then $d_i^D = d_i^I + \eta_i^G - \sigma_i$. Provided $\eta_i^G < \sigma_i$, the inline action *attenuates* the delay propagation in the network through a net reduction in the inherited delay. Similarly, $d_i^A = d_i^D + \eta_i^A$, i.e., the arrival delay is the sum of the departure delay and any randomness η_i^A encountered in the airtime due to, say, weather conditions. A dual case is illustrated for the tail $\mathcal{T}_2$. Here, f_k arrives ahead of schedule, so d_l^I, the inherited delay of f_l, is negative. The ground delay η_l^G is negative, so the flight is ready for take-off at time std_l if early departures are disallowed. However, if flight f_l is held for incoming PAX connecting from a delayed arrival flight f_j, then we have the departure delay $d_l^D = \sigma_l$, where the inline hold action *increases* the departure delay by σ_l to accommodate incoming PAX. Inline actions can thus increase or decrease the departure delays of outgoing flights.

Optimization methods: Different optimization methods used for recovery include constraint programming [6], integer programming [10, 16], mixed integer linear programming [9, 17, 25], and conic quadratic mixed integer programming [4, 7]. These exact optimization techniques do not scale for a large number of flights and hence, may not be suitable for inline recovery. Thus, heuristic approaches [8, 15, 21, 28, 29] have been developed which, though sub-optimal, can give quick solutions suitable for inline recovery. However, these heuristic approaches do not always account for uncertainty in airline operations. To incorporate such an uncertainty, the above methods may be combined with simulations [6], resulting in 'simheuristics' [13]. Digital twins may also be used to analyze, model, and optimize airline operations [5].

MaxPlus algorithm: MaxPlus [18, 27] is an algorithm for joint action selection over coordination graphs that is based on passing payoff messages along edges. It is an anytime heuristic that can scale to large graphs. It is computationally faster than similar exact methods of joint action selection like variable elimination. While it provably converges to the optimal solution for cycle-free graphs, it has also been shown empirically to work well for graphs with cycles. The benefits of this algorithm have been demonstrated in domains like multi-drone delivery [11] and urban traffic lights control [19].

3 PROBLEM STATEMENT

Figure 2 presents an overview of the problem. At each control timestep t, Optimize-Actions chooses actions $\mathbf{a}^*$ to minimize the total business cost over a horizon H ranging from t+L to t+L+H using either our approach or one of the baselines. The global impact is modeled by considering all flights of the airline departing in H across all airports. L is the lead time for implementing actions.

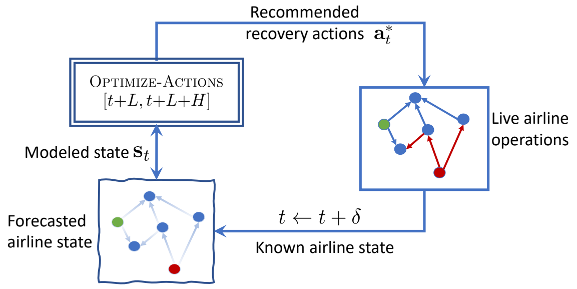


Figure 2: Overview of inline recovery

Objective: The objective function, O to be minimized is defined as a function of the actions $\mathbf{a}$ chosen for all flights to be optimized in the horizon. It is a weighed sum of the missed PAX connections, departure delays of flights, and the cost of the interventions:

$$O(\mathbf{a}) = \mathbb{E}_{\boldsymbol{\eta}}\left[w_P \sum_{ji} MC(j,i) + \sum_i w_{D_i} d_i^D\right] + \sum_i C(a_i) \quad (1)$$

$$\mathbf{a}^* = \arg\min_{\mathbf{a}} O(\mathbf{a}) \quad (2)$$

Here, w_P is the cost incurred per missed PAX (assumed constant), $MC(j,i)$ is the number of missed PAX connections from flight f_j to

f_i, w_{D_i} is the cost per unit of departure delay (assumed to depend on the airport), and $C(a_i)$ is the cost of the intervention action a_i. Because the missed connections and the delays are stochastic, we consider their expectation ($\mathbb{E}[.]$) over the intrinsic random quantities $\boldsymbol{\eta}$, namely, the random ground η_i^G and air η_i^A delays encountered by each flight f_i. The cost of actions are assumed deterministic. The key constraints are as follows:

$$\forall t'_{t' \in [t, t+H]} \forall k \left(\sum_{i, t' \in tap_i} a_{ik} \leq R_k \right) \tag{3}$$

$$\forall i, b_i = sta_j + d_j^A \tag{4}$$

$$\forall i, d_i^I = d_j^A \tag{5}$$

$$\forall i, d_i^D = \max(\max(d_i^I + \eta_i^G - buf_i, 0) + \sigma(a_i), 0) \tag{6}$$

$$\forall i, e_i = std_i + d_i^D \tag{7}$$

$$\forall i, d_i^A = d_i^D + \eta_i^A \tag{8}$$

Resource constraints: Constraint 3 ensures that at any time, the number of recovery resources used by all flights f_i for each type of action k does not exceed the resource count R_k at each airport. Specifically, if f_i chooses action k (making the binary variable $a_{ik} = 1$), a resource of type k would be used during the entire turn-around process duration tap_i of f_i. Note that this resource constraint is only enforced for flights departing from the same airport and have potential overlaps in their turn-around processes.

Delay propagation constraints: Constraint 4 states that a flight f_i starts its turn-around (at b_i) when the previous flight f_j of the same tail arrives, which in turn is at the sum of the scheduled time of arrival sta_j and its arrival delay d_j^A. Constraint 5 makes the inherited delay of a flight d_i^I equal the arrival delay of its previous flight d_j^A. Constraint 6 relates the departure delay d_i^D to the inherited delay; random ground-delay η_i^G; the built-in schedule buffer buf_i; and the delay change $\sigma(a_i)$ due to action a_i for flight f_i. The double max formulation disallows early departures. Constraint 7 relates the end-time of service e_i to the scheduled time of departure std_i and the departure delay d_i^D. Constraint 8 relates the arrival delay of a flight d_i^A to its departure delay and the random air-delay η_i^A.

3.1 Stochastic Approximation Objective

Constraints 6 and 8 explicitly include external noise from ground and air operations as in reality. Because the objective has components that are non-linear functions of delays (e.g., PAX miss or do not miss a connection), obtaining a closed form expression for the expectation in the objective is non-trivial for an arbitrary initial state of the network. Instead, we approximate the expectation of the objective as a sample average across several evolutions of the network (i.e., *forecasted optimization scenarios*) and obtain *one common* action vector that optimizes the sample average. This is commonly referred to as *scenario-based optimization* in literature. Specifically, we obtain the actions to be chosen as $a^* = \arg\min_{\mathbf{a}} \hat{O}(\mathbf{a})$ where

$$O(\mathbf{a}) \approx \hat{O}(\mathbf{a}) \triangleq \frac{\sum_{k=1}^{k=N} O(\mathbf{a}, \eta_k)}{N} \tag{9}$$

for N forecasted optimization scenarios. Once η_k are sampled, the optimization problem for $\hat{O}$ can be solved using solvers. However, as we show later, conventional optimizers can be too time consuming to solve the problem in real-time for inline recovery.

4 SOLUTION APPROACH

We now describe Stochastic-Minplus-with-State (SMS) — our agent-based approach for inline recovery. In MaxPlus, agents are nodes in a *coordination graph*. Each agent exchanges messages with its neighbors in the *coordination graph* to jointly coordinate their actions to maximize a common global objective. The common global objective is defined using node-level ($\phi_i(a_i)$) and edge-level ($\phi_{ij}(a_i, a_j)$) deterministic cost functions of agent actions a_i, a_j as

$$\sum_{v_i \in V} \phi_i + \sum_{e_{ij} \in E} \phi_{ij} \tag{10}$$

Our approach: In SMS, we generalize MaxPlus algorithm for inline recovery. Specifically, each flight in our approach is an agent and a coordination graph is defined using domain knowledge to model the interactions between flights. In addition, we include a *state* for each agent and decide coordinated actions in a *stochastic* environment. Thus, our approach allows state-dependent cost functions as opposed to purely action-dependent cost functions. Because we allow randomness, our approach can handle expectations as a joint objective rather than a deterministic function. The *state-dependent* cost functions ϕ_i' and ϕ_{ij}' are obtained by evaluating the ϕ functions in the original algorithm for specific states s_i and s_j as $\phi_i(a_i; s_i)$ and $\phi_{ij}(a_i, a_j; s_i, s_j)$ As the state of a flight depends on the actions taken by a previous flight, we use a *fixed point-iteration* for state-updates between each message passing round.

4.1 Coordination Graph for Airline Operations

We start by constructing an undirected coordination graph $G = (V, E)$. The vertex set V is the set of flights F departing in the optimization window (of length H). An undirected edge e_{ij} is added between flights f_i and f_j in G if f_i and f_j share either the aircraft or connecting PAX. Specifically, a *tail edge* is added to the graph if the same physical aircraft serving as incoming flight f_j departs as outgoing flight f_i. In this case the arriving delay d_j^A of flight f_j becomes the inherited delay d_i^I of flight f_i during state updates. A *PAX edge* is added to the graph if PAX on incoming flight f_k connect to outgoing flight f_i. So a delay of f_k would influence the value of connections missed by f_i depending upon the arrival delay of f_k and the departing delay of f_i. We allow a tail parent of a flight to also be a PAX parent if needed with an adjusted cost function. Because the tail-plan and the PAX connections are assumed to be known, we treat the *structure* of the coordination graph as stationary in each window. The extent of influence would, however, vary with the individual inherited, arrival, and departure delays of the flights, and the number of PAX connections.

4.2 State-Dependent Cost Functions

State: The state of each node (flight) f_i in the graph is the *inherited delay* d_i^I. We refer to nodes who have no neighbors departing ahead of them in time in the graph as *sources*. The *observed* state d^I of the source nodes is obtained from the operational data as the result of the implemented actions of the previous decision windows. For the non-source nodes (whose actions are to be decided), the actions chosen as the algorithm progresses can be propagated through the coordination graph to obtain a *forecasted* state. This is captured in the procedure Update-State which uses Equations 4 – 8 to update the forecasted state of the network.

Cost functions: The cost functions of nodes and edges in the coordination graph are such that the sum of the node costs and the edge costs defined in the MaxPlus objective in Equation 10 matches exactly with the business optimization objective mentioned in Equation 1. We expect the cost of an action a_i of flight f_i ($C_i(a_i)$) to be the node cost of flight f_i. However, for numerical stability, we choose *node cost* to be zero; and instead, amortize the action cost across all edges incident on the node as $c'_i(a_i) \triangleq C_i(a_i)/\Delta_i$ where Δ_i is the degree of f_i in G. Given the states d^I_i and d^I_j of the flights in an edge e_{ij} in G, we define the *edge cost* functions for the actions as follows. For a tail edge e_{ij}, the cost of the edge is the average weighted departure delay of the arriving and departing flights f_i and f_j, i.e., $\phi'_{ij}(a_i, a_j, d^I_i, d^I_j) \triangleq (w_{D_i} d^D_i + w_{D_j} d^D_j)/2 + c'_i(a_i) + c'_j(a_j)$. Because any non-boundary-condition flight occurs as part of two tail edges, we divide the delay cost by 2. For boundary-condition flights, we add the departure delay cost without dividing by 2. For a PAX edge e_{ij}, the cost $\phi'_{ij}(a_i, a_j, d^I_i, d^I_j) \triangleq w_P MC(i, j) + c'_i(a_i) + c'_j(a_j)$, i.e., the cost of the PAX who miss connections given the inherited delays of the flights and the actions they take. With these definitions, the objective in our SMS algorithm matches Equation 1. This cost functions determination procedure, named Get-Cost, is used in lines 3, 22, and 24 of the SMS Algorithm 1.

4.3 Algorithm SMS

Given the current state of the network in the coordination graph G, and an initial set of actions, SMS begins by initializing the graph with an *estimated* state for all flights in the optimization window. At the end of this step in line 2 in Algorithm 1, each flight f_i has an actual or estimated inherited delay d^I_i as the result of the initial set of actions. Collectively the state of the network is represented as the set of all inherited delays $\mathbf{d}^I$. The local cost functions ϕ'_{ki} and ϕ'_{kij} are initialized in line 3 using the Get-Cost procedure. As with MaxPlus, SMS proceeds in rounds till convergence. However, we additionally consider stochastic optimization scenarios as in Equation 9. These scenarios are parametrized as η_k. For each η_k, independent random samples of all unseen disturbances are drawn from the ground delay and the air delay distributions. As in the baseline IP and CP approaches, the objective is to identify the optimal actions that minimize the sample averaged approximation of the expected objective. Therefore, the actions for all flights are the same across all the scenarios. In each round, a flight f_i coordinates with its neighbors f_j to decide the action a^*_i that is the most beneficial at a global level. To this end, in lines 6 through 10, each flight f_i sends a message μ_{kij} to f_j that is defined as follows:

$$\mu_{kij}(a_j) \triangleq \min_{a_i} \left(\phi'_{ki}(a_i) + \phi'_{kij}(a_i, a_j) + \sum_{l \in \mathcal{N}_i - \{f_j\}} \mu_{kli}(a_i) \right) \quad (11)$$

The terms in Equation 11 also depend on the appropriate states d^I and the scenario η_k but we omit them for brevity. Intuitively, through message μ_{kij}, the agent f_i informs its neighbor f_j (in scenario η_k) about the effect of the action a_j chosen by f_j when considering the local cost at f_i (using ϕ'_{ki}), the pairwise edge cost of

Algorithm 1: Stochastic-MinPlus-with-State (G)

```
     // Compute projected state of network assuming all
     // flights choose no intervention action
1    $\mathbf{a}^*$ = NOOP
2    $\mathbf{d}^I$ = Update-State ($G, \mathbf{a}^*$)
3    $\forall i, j, k\ \phi'_{ki} = 0, \phi'_{kij}$ = Get-Cost($G, \mathbf{d}^I, \mathbf{a}^*$)
4    converged = False
5    while not converged
         // Compute payoff messages $\mu_{kij}(a_j; \eta_k)$ for each
         // scenario $\eta_k$ using only actions while
         // assuming the state does not change with action
         // chosen
6        for each stochastic scenario $\eta_k$
7            for each flight $f_i$ in $G$
8                for each $f_j$ in neighbors $\mathcal{N}(f_i)$ of $f_i$
9                    for each action $a_j$ of $f_j$
10                       Compute $\mu_{kij}(a_j; \eta_k)$ as per Eq. 11
         // Compute the local payoff at each node $f_i$
         // depending on how it affects its neighbors
11       for each stochastic scenario $\eta_k$
12           for each flight $f_i$ in $G$
13               for each action $a_i$ of $f_i$
14                   $g_{ki}(a_i; \eta_k) = \phi'_{ki}(a_i; \eta_k) + \sum_{j \in \mathcal{N}_i} \mu_{kji}(a_i; \eta_k)$
         // Choose optimal action for flight $f_i$
15       for each flight $f_i$ in $G$
16           $a^*_i = \arg\min_{a_i} \sum_k g_{ki}(a_i; \eta_k)$
         // Enforce resource constraints
17       for each flight $f_i$ in $G$
18           $a^*_i$ = Resolve-Conflicts($G, \mathbf{a}^*, \mathbf{d}^I$)
         // Update state to be in sync with actions
         // chosen in current round
19       $\mathbf{d}^I$ = Update-State($G, \mathbf{a}^*$)
         // Compute state-dependent cost function
20       for each stochastic scenario $\eta_k$
21           for each flight $f_i$ in $G$
22               $\phi'_{ki}(a_i, d^I_i; \eta_k) = 0$
23               for each neighbor $f_j$ of $f_i$
24                   $\phi'_{kij}(a_i, a_j, d^I_i, d^I_j; \eta_k)$ =
                         Get-Cost($G, \mathbf{d}^I, \mathbf{a}^*$)
```

the interaction between f_i and f_j (using ϕ'_{kij}), and the effects of neighbors f_l (in neighborhood $\mathcal{N}_i$) *other than* f_j (recursively using previous estimates of μ_{kli}). Once an agent f_i receives the μ messages from all its neighbors f_j, it consolidates the local payoff of its (i.e., f_i's) actions through the g_{ki} function in line 14.

Stochastic averaging: The μ and g functions are computed by each agent f_i for every stochastic scenario η_k. However, the optimal action a^*_i for f_i is chosen as that which minimizes the ensemble averaged g function in line 16. This is in line with our goal of minimizing an expectation approximation as in Equation 9.

Resource constraints: We handle resource constraints by identifying and resolving any conflicts in procedure Resolve-Conflicts across flights departing from the same airport with overlapping turn-arounds. Flights are ordered by increasing $\sum_k g_{ki}(a^*_i; \eta_k)-$

Table 1: Cost and benefit of actions

Action	Hub cost	Spoke cost	Delay change (min)
NOOP	0	0	0
PARALLELIZE	600	400	-5
SURGE	1200	800	-10
HOLD_5 min	60	30	+5
HOLD_10 min	120	60	+10
HOLD_15 min	180	80	+15

$\sum_k g_{ki}(\text{NOOP}; \eta_k)$, where a_i^* is the optimal action chosen in line 16. Flights appearing earlier in this ordering get preference on the shared resource. If no resource is free during the turn-around of a flight, its optimal action is flipped to the next-best action given by its g function in line 14. This process is repeated for all the flights and all the intervention actions till there are no more action flips. At the end of RESOLVE-CONFLICTS, we get finalized optimal actions in line 18 that adhere to the resource constraints as per Equation 3.
State and costs update: In SMS, an agent's action (e.g., tail-parent's hold action) influences the state of another agent (e.g., the inherited delay of the tail-child). To model this, we implement UPDATE-STATE in line 19 that propagates the effect of all currently chosen actions across the entire network using the coordination graph to give a new estimated state. Similarly, the local cost functions ϕ' are updated for specific states in lines 20 to 24 using the GET-COST procedure. To avoid numerical instabilities, we normalize the payoff functions μ by the mean across all the actions as in MAXPLUS.

5 EXPERIMENT SETUP

Dataset: We obtained the tail-plans and schedules of a leading North American carrier for over a period of 1 week. The tail-plans provide the mapping between physical aircraft and logical flights in the airline network. The schedule provides the details of the scheduled times of arrival and departure of the flights. We assume that an aircraft on the ground that has a significant gap between the incoming and outgoing flights is tugged out to the hangar or tarmac as is the conventional practice. Such an aircraft will be considered for inline recovery only if it is extremely delayed. For PAX data, we could obtain only representative data from the airline due to privacy concerns. Therefore, we generate synthetic PAX data from the BTS website using the itineraries available. Specifically, we assume that connections are possible between an incoming and outgoing flight at an airport if the separation between them is higher than a minimum connection time of 45 minutes. Next, we assign a PAX on the incoming flight to a connecting itinerary with a probability determined by the weight of the OD pairs in the BTS database. This is done while ensuring that all incoming and outgoing PAX are accounted for including local ground arrivals and departures. Table 1 summarizes the costs and benefits (delay changes) of the actions for hub and spoke airports [23].
Simulation environment: We implement a simulation environment for airline operations in `simpy` [3], a discrete event simulation library based in `Python`. Using our dataset curated from real-world sources, we simulate the arrival, turn-around, and departure processes of an airline network at scale, with random ground and air delays drawn from uniform distributions with parameters [0, 10] minutes and [-4, 8] minutes, respectively. We assume a lead time L of 30 minutes, and a control timestep δ of 45 minutes. At each airport, we model resource contention to ensure that an arbitrary number of flights cannot access recovery resources for an inline action at the same time. We validate the simulation environment by comparing the average delay, on-time performance, and the number of PAX missing connections with the real-world environment.
Baselines: We use the following as four baselines for SMS:

- **NOOP**: We proceed with regular operations without any operational intervention. This serves as a bound for the performance on business metrics.
- **GREEDY**: Each flight makes a *locally* optimal decision to optimize the objective in Equation 1 just for that flight alone. This serves to quantify the difference between considering a purely local vs a global objective.
- **CP (IP)** The objective in Equation 1 is solved for a window of optimization using the constraint (integer) programming solver `cp-sat (SCIP)` from the Google `or-tools` optimization suite [22].

Operational Regimes: We consider the following two regimes:

- **BAU:** This represents business-as-usual evolution of the network delays without any irregular operations.
- **IROP:** This represents interrupted operations where all departing flights at airports in the same geographical area incur higher delays due to temporary irregular operations.

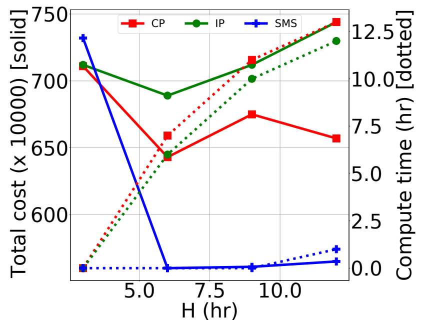
Scenarios: For algorithms that use scenario-based optimization, we consider $|\eta_k|$=10 stochastic forecasted optimization scenarios. Each regime is evaluated as an average of 20 stochastic evaluation scenarios that are independent of the optimization scenarios.
Performance metrics: All algorithms are run with a maximum timeout of 20 minutes reflective of the typical control time available for airline operations. The business metrics considered are: 1) the average cost of departure delay of flights (assuming representative w_D of 180 and 60 units/minute at hub and spoke airports, respectively); 2) the average cost of missed PAX connections (assuming w_P = 200); 3) the average cost of the actions chosen for inline interventions; and 4) a total cost that is a combination of all these. A lower cost is better. If the algorithm completes within the timeout, we consider the optimal solution, else we consider the best intermediate solution. In addition, we consider the run-time of algorithms to either completion or timeout on a server-class machine.

6 RESULTS

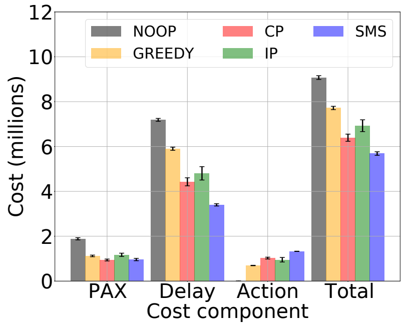
Optimization horizon H: Figure 3a shows the rationale for the choice of H. A larger H would capture more global effects but take more time. The total cost initially decreases with increasing H. It saturates at H = 6 hours for SMS and increases for others due to the inability to find good solutions. However, the total compute time across all decision windows, as expected, keeps increasing with H. Therefore, we report our results for H = 6 hours for all algorithms.

6.1 Aggregate Performance

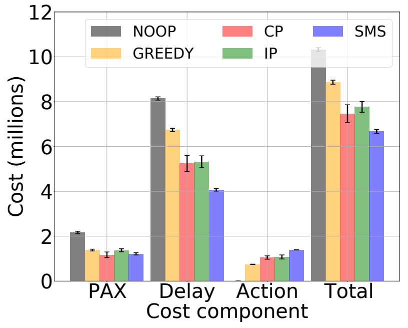
Figures 3b and 3c show the business metrics, averaged (by mean) over the evaluation scenarios, for BAU and IROP regimes, respectively. The X-axis shows individual components of the business



(a) Choice of horizon H

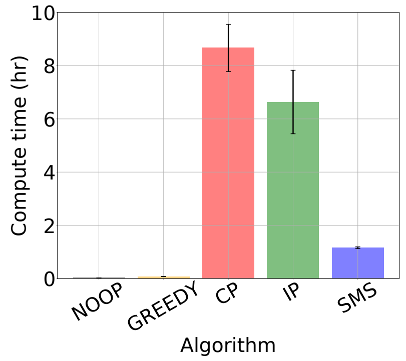


(b) BAU

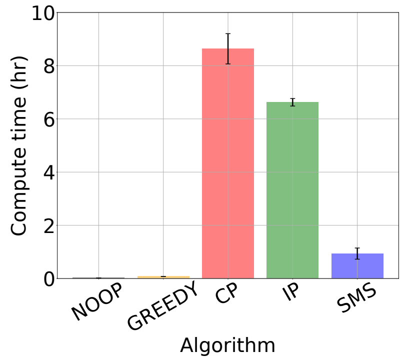


(c) IROP

Business metrics



(d) BAU



(e) IROP

Compute times of algorithms

Figure 3: Aggregate performance (best viewed in color; error bars represent ± 1 standard deviation)

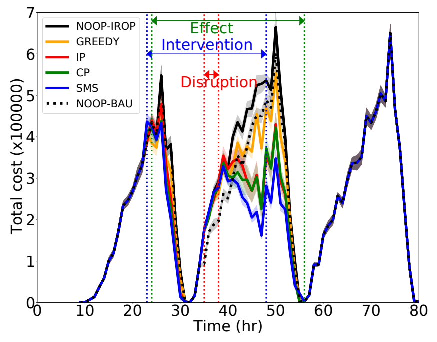
metrics and their combination. The Y-axis shows cost for each component. We observe the following. First, NOOP in IROP is worse by 14% than NOOP in BAU in terms of the total cost, confirming that inline recovery is more important during interrupted operations. Second, the *locally optimal* GREEDY *strategy is sub-optimal at a global level*. In terms of the total cost, SMS outperforms GREEDY by 26.3% (24.7%) for BAU (IROP), confirming a need to consider higher order global effects. Third, with a timeout for decisions, for both regimes (BAU, IROP), *SMS improves total cost* over IP (by 17.9%, 14.1%) and CP (by 11.0%, 10.6%). Figures 3d and 3e show the total compute time across all optimization windows, averaged (by median) over the evaluation scenarios, for the BAU and IROP regimes, respectively. IROP roughly takes the same time as BAU (e.g., 2% difference for IP). The simulation time (to implement the optimizer decisions) is roughly constant around 90 seconds for each algorithm. However, the decision time ranges from 60 milliseconds for GREEDY to around 9 hours for CP. For most decision windows, CP and IP hit the compute timeout without reaching optimality.

Optimality gap: When IP and CP are run without a timeout, they do not give an optimal solution for any $H > 3$ hours even after several days of computation for the complete case of 10 optimization and 20 evaluation scenarios. Hence, we investigate the optimality gap on simpler instances. For $H = 3$ hours, we find that SMS is within 5% of the optimal solution obtained by CP for the complete case. For $H = 6$ hours with 1 optimization and 20 evaluation scenarios, we find that SMS is within 4% of the optimal solution from IP.

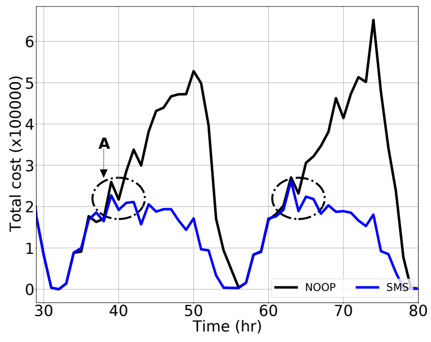
Reproducibility: We re-evaluate the algorithms for their aggregate performance in the BAU regime on another airline with a different topology. For the new airline, SMS outperforms GREEDY (CP) by 15% (9.1%) in terms of the total cost. SMS is also better by 13x in computation time over CP with timeout. As these improvements are similar to those observed for the original airline, the performance of the SMS algorithm is reproducible on different airlines. We omit detailed results and plots for the new airline for the sake of brevity.

6.2 Dynamic Performance

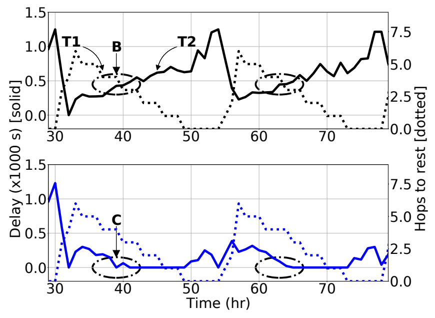
Figure 4a shows the time evolution of the algorithms for the IROP regime. Due to the cyclical nature of airline operations, we see a cyclical behavior in the cost. The X-axis shows the hour t of operation. The Y-axis shows the total cost for all flights departing between t and t+1 hours. Each curve shows the ensemble mean over 20 evaluation scenarios and the associated error regions. SMS and other control interventions are implemented from $t = 23$ to $t = 48$ hours; and the effects are seen from $t = 24$ to $t = 56$ hours. The IROP regime commences at $t = 35$ and ends at $t = 38$ hours. The curve 'NOOP-IROP' shows the cost with these IROP and no intervention. As a counterfactual, the curve 'NOOP-BAU' shows the cost *without IROP (i.e., BAU)* and no intervention. NOOP-IROP and NOOP-BAU diverge around $t = 33$ hours showing the additional cost due to IROP. Till $t = 24$ hours, there is no effect of the intervention due to lead-time. At $t = 24$ hours, the effect of the inline recovery algorithms kicks in; and we observe differing performance. GREEDY shows



(a) Business cost



(b) BAU: Inline recovery viability



(c) BAU: Sweet-spot for inline recovery

Figure 4: Dynamic performance of the algorithms (best viewed in color)

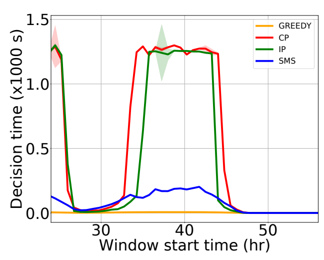
the least improvement; while SMS has the maximum improvement. Because global effects compound over time, the inability of CP/IP to compute good solutions within a timeout initially could potentially limit them to trajectories with higher cost.

6.3 Viability of Inline Recovery

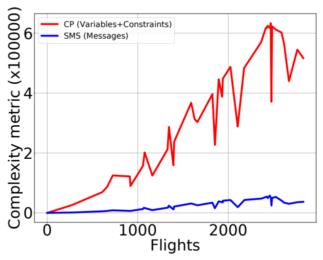
Figure 4a shows that the benefits of inline recovery occur during the region marked 'Effect'. *The effect is the most pronounced after the point marked* ***A*** *in Figure 4b.* We consider the BAU scenario with NOOP and with SMS in Figure 4b for an extended intervention and simulation period. We see control helps only after the same *phase of cyclical operations* marked **A** in Figure 4b. To explain this phenomenon, we consider a metric called *hops to rest.* Delays build up as an aircraft takes multiple hops during the day as flights across multiple airports before it goes to rest at a designated airport. A rest stop is a buffer in the schedule that is large enough to absorb any delay. Hops to rest is therefore the number of future flights an aircraft makes before resting. Figure 4c shows the average departure delay and average hops to rest as a function of the departure hour. The top-panel shows NOOP while the bottom panel shows SMS. As an aircraft accumulates delays across flight hops, it moves closer to its rest stop, so hops to rest decreases while delay increases. Intuitively, when hops to rest is low, even a highly delayed flight will have less global impact (trend **T1**). Conversely, when hops to rest is high, delays are likely so less that they will not have much global impact due to schedule buffers (trend **T2**). Therefore, *intervention becomes viable when the opposing network effects* ***T1*** *and* ***T2*** *form a sweet-spot.* This is exactly the points marked **B** for NOOP and **C** for SMS, which also correspond to **A** in Figure 4b.

6.4 Scalability of SMS

Figure 5a shows the decision time for each window. GREEDY and NOOP are insensitive to the number of flights because they take purely local decisions. The decision times for CP and IP increase more rapidly than SMS when the number of flights is large. This can be explained by considering the internals of the algorithms. Figure 5b shows two curves: 1) the number of variables and constraints for CP; and 2) the number of messages exchanged by SMS. While these two are not directly comparable, we empirically observe that *SMS scales approximately linearly with increasing problem sizes (number of flights)*, unlike CP which scales non-linearly. This is because each flight node in the coordination graph likely has a bounded degree due to interacting edges with other flights, and thus a roughly linear upper bound on the number of edges. Hence, SMS outspeeds classical optimization even while producing reasonable .



(a) Dynamic decision time



(b) Internal complexity

Figure 5: Scalability of SMS (best viewed in color)

6.5 Limitations

First, our approach models delays as uniformly distributed; in reality they could be long-tailed. Second, our quantitative approach may not directly handle qualitative, subjective priorities of an airline (e.g., a flagship flight that cannot be delayed). These need to be translated to an objective cost which may not be possible. Third, while agent-based coordination is light-weight, it is not guaranteed to converge always and we may need a fallback heuristic mechanism. Fourth, our optimization approach does not consider variance and we could improve it by minimizing the objective while bounding variance.

7 CONCLUSIONS

We addressed the problem of inline recovery with a scalable agent-based algorithm SMS that can consider higher order global effects. Our evaluation shows that SMS scales better than conventional optimization approaches even when achieving similar or better solution quality in less compute time. Given appropriate data, we expect our algorithm to be reproducible and show similar improvements for different airlines. We are currently discussing with our airline business customer for a trial deployment on their development environment. Directions for future work include considering explicit crew constraints, larger action space, various aircraft types, and variance of business objectives in the optimization.

REFERENCES

[1] [n.d.]. Bureau of Transportation Statistics: Airline On-Time Statistics and Delay Causes. https://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp. Accessed: 2023-10-09.

[2] [n.d.]. Federal Aviation Administration: Cost of Delay Estimates 2019. https://www.faa.gov/sites/faa.gov/files/data_research/aviation_data_statistics/cost_delay_estimates.pdf. Accessed: 2023-10-09.

[3] [n.d.]. *SimPy*. https://simpy.readthedocs.io/en/latest/contents.html

[4] M. Selim Aktürk, Alper Atamtürk, and Sinan Gürel. 2014. Aircraft Rescheduling with Cruise Speed Control. *Operations Research* 62, 4 (2014), 829–845. http://www.jstor.org/stable/24540664

[5] Michael Arguello, Charles P Ephraim, Jonathan Mark Dunsdon, Stephen Jonathan Davis, and Kristin M Schanche. 20 April 2023. Network Digital Twin of Airline Operations. https://patents.google.com/patent/US20230118644A1. Pub. no. US 2023/0118644 A1 (20 April 2023).

[6] P. Arias, M. Mota, D. Guimarans, and G. Boosten. 2013. A Methodology Combining Optimization and Simulation for Real Applications of the Stochastic Aircraft Recovery Problem. In *2013 8th EUROSIM Congress on Modelling and Simulation (EUROSIM)*. IEEE Computer Society, Los Alamitos, CA, USA, 265–270. https://doi.org/10.1109/EUROSIM.2013.55

[7] Uğur Arıkan, Sinan Gürel, and M. Selim Akturk. 2017. Flight Network-Based Approach for Integrated Airline Recovery with Cruise Speed Control. *Transp. Sci.* 51 (2017), 1259–1287. https://api.semanticscholar.org/CorpusID:13831195

[8] Serge Bisaillon, Jean-François Cordeau, Gilbert Laporte, and Federico Pasin. 2011. A large neighbourhood search heuristic for the aircraft and passenger recovery problem. *4OR* 9 (2011), 139–157.

[9] Srinivas Bollapragada, Nitish Umang, Sanket Bhat, Hocine Bouarab, and Marc A Garbiras. 21 July 2022. Methods and Systems for Generating Holistic Airline Schedule Recovery Solutions Accounting for Operations, Crew, and Passengers. https://patents.google.com/patent/US20220230108A1. Pub. no. US 2022/0230108 A1 (21 July 2022).

[10] Jens O. Brunner. 2014. Rescheduling of flights during ground delay programs with consideration of passenger and crew connections. *Transportation Research Part E: Logistics and Transportation Review* 72 (2014), 236–252. https://doi.org/10.1016/j.tre.2014.10.004

[11] Shushman Choudhury, Jayesh K. Gupta, Peter Morales, and Mykel J. Kochenderfer. 2021. Scalable Anytime Planning for Multi-Agent MDPs. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (Virtual Event, United Kingdom) *(AAMAS '21)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 341–349.

[12] Jan Evler, Martin Lindner, Hartmut Fricke, and Michael Schultz. 2022. Integration of turnaround and aircraft recovery to mitigate delay propagation in airline networks. *Computers & Operations Research* 138 (2022), 105602. https://doi.org/10.1016/j.cor.2021.105602

[13] Yagmur S. Gök, Silvia Padrón, Maurizio Tomasella, Daniel Guimarans, and Cemalettin Ozturk. 2023. Constraint-based robust planning and scheduling of airport apron operations through simheuristics. *Annals of Operations Research* 320, 2 (January 2023), 795–830. https://doi.org/10.1007/s10479-022-04547-

[14] L.K. Hassan, B.F. Santos, and J. Vink. 2021. Airline disruption management: A literature review and practical challenges. *Computers & Operations Research* 127 (2021), 105137. https://doi.org/10.1016/j.cor.2020.105137

[15] Yuzhen Hu, Yan Song, Kang Zhao, and Baoguang Xu. 2016. Integrated recovery of aircraft and passengers after airline operation disruption based on a GRASP algorithm. *Transportation Research Part E: Logistics and Transportation Review* 87 (2016), 97–112. https://doi.org/10.1016/j.tre.2016.01.002

[16] Yuzhen Hu, Baoguang Xu, Jonathan F. Bard, Hong Chi, and Min'gang Gao. 2015. Optimization of multi-fleet aircraft routing considering passenger transiting under airline disruption. *Computers & Industrial Engineering* 80 (2015), 132–144. https://doi.org/10.1016/j.cie.2014.11.026

[17] Niloofar Jafari and Seyed Hessameddin Zegordi. 2011. Simultaneous recovery model for aircraft and passengers. *Journal of the Franklin Institute* 348, 7 (2011), 1638–1655. https://doi.org/10.1016/j.jfranklin.2010.03.012 Special issue on Modeling, Simulation and Applied Optimization.

[18] Jelle R. Kok and Nikos Vlassis. 2006. Collaborative Multiagent Reinforcement Learning by Payoff Propagation. *Journal of Machine Learning Research* 7 (Dec 2006), 1789–1828.

[19] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. 2008. Multiagent Reinforcement Learning for Urban Traffic Control Using Coordination Graphs. In *Machine Learning and Knowledge Discovery in Databases*, Walter Daelemans, Bart Goethals, and Katharina Morik (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 656–671.

[20] Tejasvi Malladi, Karpagam Murugappan, Depak Sudarsanam, Ramasubramanian Suriyanarayanan, and Arunchandar Vasan. 2021. To Hold or Not to Hold? - Reducing Passenger Missed Connections in Airlines Using Reinforcement Learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (Virtual Event, United Kingdom) *(AAMAS '21)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 862–870.

[21] C Mancel N Jozefowiez and F Mora-Camino. 2013. A heuristic approach based on shortest path problems for integrated flight, aircraft, and passenger rescheduling under disruptions. *Journal of the Operational Research Society* 64, 3 (2013), 384–395. https://doi.org/10.1057/jors.2012.20 arXiv:https://doi.org/10.1057/jors.2012.20

[22] Laurent Perron and Vincent Furnon. [n.d.]. *OR-Tools*. Google. https://developers.google.com/optimization/

[23] Everett B. Peterson, Kevin Neels, Nathan Barczi, and Thea Graham. 2013. The Economic Cost of Airline Flight Delay. *Journal of Transport Economics and Policy* 47, 1 (2013), 107–121. http://www.jstor.org/stable/24396355

[24] Judith Rosenow, Philipp Michling, Michael Schultz, and Jörn Schönberger. 2020. Evaluation of Strategies to Reduce the Cost Impacts of Flight Delays on Total Network Costs. *Aerospace* 7, 11 (2020). https://doi.org/10.3390/aerospace7110165

[25] Bruno F. Santos, Maarten M.E.C. Wormer, Thomas A.O. Achola, and Richard Curran. 2017. Airline delay management problem with airport capacity constraints and priority decisions. *Journal of Air Transport Management* 63 (2017), 34–44. https://doi.org/10.1016/j.jairtraman.2017.05.003

[26] Yi Su, Kexin Xie, Hongjian Wang, Zhe Liang, Wanpracha Art Chaovalitwongse, and Panos M. Pardalos. 2021. Airline Disruption Management: A Review of Models and Solution Methods. *Engineering* 7, 4 (2021), 435–447. https://doi.org/10.1016/j.eng.2020.08.021

[27] N. Vlassis, R. Elhorst, and J.R. Kok. 2004. Anytime Algorithms for Multiagent Decision Making Using Coordination Graphs. In *International Conference on Systems, Man, and Cybernetics*. The Hague, The Netherlands.

[28] Tianshun Yang and Yuzhen Hu. 2019. Considering Passenger Preferences in Integrated Postdisruption Recoveries of Aircraft and Passengers. *Mathematical Problems in Engineering* 2019 (2019).

[29] Seyed Hessameddin Zegordi and Niloofar Jafari. 2010. Solving the Airline Recovery Problem By Using Ant Colony Optimization. *International Journal of Industrial Engineering and Production Research* 21, 3 (2010), 121–128.