Transportation Research Part E 179 (2023) 103295

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Towards efficient airline disruption recovery with reinforcement learning

Yida Ding [a], Sebastian Wandelt [b], Guohua Wu [c], Yifan Xu [b], Xiaoqian Sun [b,*]

[a] School of Vehicle and Mobility, Tsinghua University, 100084 Beijing, China
[b] School of Electronic and Information Engineering, Beihang University, 100191 Beijing, China
[c] Department of Information Management, Air China, Beijing, China

ARTICLE INFO

Keywords:
Airline scheduling
Disruptions
Deep Reinforcement Learning

ABSTRACT

Disruptions to airline schedules precipitate flight delays/cancellations and significant losses for airline operations. The goal of the integrated airline recovery problem is to develop an operational tool that provides the airline with an instant and cost-effective solution concerning aircraft, crew members and passengers in face of the emerging disruptions. In this paper, we formulate a decision recommendation framework which incorporates various recovery decisions including aircraft and crew rerouting, passenger reaccommodation, departure holding, flight cancellation and cruise speed control. Given the computational hardness of solving the mixed-integer nonlinear programming (MINP) model by the commercial solver (e.g., CPLEX), we establish a novel solution framework by incorporating Deep Reinforcement Learning (DRL) to the Variable Neighborhood Search (VNS) algorithm with well-designed neighborhood structures and state evaluator. We utilize Proximal Policy Optimization (PPO) to train the stochastic policy exploited to select neighborhood operations given the current state throughout the Markov Decision Process (MDP). Experimental results show that the objective value generated by our approach is within a 1.5% gap with respect to the optimal/close-to-optimal objective of the CPLEX solver for the small-scale instances, with significant improvement regarding runtime. The pre-trained DRL agent can leverage features/weights obtained from the training process to accelerate the arrival of objective convergence and further improve solution quality, which exhibits the potential of achieving Transfer Learning (TL). Given the inherent intractability of the problem on practical size instances, we propose a method to control the size of the DRL agent's action space to allow for efficient training process. We believe our study contributes to the efforts of airlines in seeking efficient and cost-effective recovery solutions.

1. Introduction

Deviation from an optimized airline schedule leads to significant losses for airlines. While causes for such deviations are manifold, we discuss a few of these here. One of the major drivers of airline delay and cancellations is adverse weather conditions, which prevent aircraft to be operated in a normal environment. A second factor for disruption is congestion, particularly at hub airports operated under tight operational constraints. Such congestions may appear across the entire terminal layout, including passenger

* Corresponding author.
E-mail addresses: ding001021@gmail.com (Y. Ding), wandelt@informatik.hu-berlin.de (S. Wandelt), wuguohua@airchina.com (G. Wu), yifan_buaa@buaa.edu.cn (Y. Xu), sunxq@buaa.edu.cn (X. Sun).

https://doi.org/10.1016/j.tre.2023.103295
Received 28 June 2023; Received in revised form 23 August 2023; Accepted 23 September 2023
Available online 5 October 2023
1366-5545/© 2023 Elsevier Ltd. All rights reserved.

security lanes, boarding, runway congestion, and others. Third, human and mechanical failures can lead to an inability to operate aircraft as planned. For instance, such cases include the sudden unavailability of sufficient crew members for a flight or the malfunction of parts of an aircraft. These examples for disruption scenarios can all be described as rather local by initial effect. Nevertheless, the ultimate extent of such a local phenomenon can reach much beyond the initial scale, as failures and delay easily propagate downstream in the aviation system. This propagation can possibly be explained by the combination of two major factors. First, most traditional airlines operate according to a hub-and-spoke model, where aircraft are routed through a set of centralized airports in a small number of cities. Such aggregation leads to so-called economies of scale and allows airlines to cost-effectively serve a wider range of markets (O'Kelly and Bryan, 1998; Campbell and O'Kelly, 2012). The presence of these single points of failure leads to a high degree of vulnerability (O'Kelly, 2015). Given that hubs are under high operational pressure and have significant potential for congestion, the system is inherently susceptible to disruptions. Second, given the rising fuel cost, increase in competition from low-cost carriers, and various other adverse ramifications, the airline business is under high economic pressure (Dennis, 2007; Zuidberg, 2014). Accordingly, airline schedules are typically highly optimized with the goal to maximize the airline's profit under ideal conditions. Deviations from these ideal conditions inevitably lead to mismatches across the global usage of airline's resources.

Given that disruptions are extremely costly for airlines, with an estimated cost of 60 billion per year — equivalent to eight percent of the worldwide airline revenue (Serrano and Kazda, 2017), recent studies on airline scheduling have increasingly investigated the problems coming with disruptions (Kohl et al., 2007; Clausen et al., 2010; Hu et al., 2016; Rapajic, 2018; Wen et al., 2020, 2022). Conceptually, disruption can be divided into two phases from the perspective of an airline. First, the emergency-like event of the disruption taking place and the initial response of an airline. For example, such an event can be the closure of a runway, which reduces the airport's capacity if multiple runways are present. During the emergence phase, the impact of the disruptions begins to turn from local to global. Accordingly, this phase is followed by a time of recovery, in which the airline makes decisions to reduce the overall impact on the airline and its passengers, involving passenger rebooking, aircraft rescheduling, and other actions.

In this study, we formulate a problem for integrated airline recovery and perform a thorough investigation on how to solve the problem effectively for large-scale instances. The underlying model is expressive in being able to incorporate aircraft and crew rerouting, passenger reaccommodation, departure holding, flight cancellation, while considering aircraft cruise speed decisions. Given the computational hardness of the model, we develop a set of solution techniques. First, following the existing literature on related problems, we develop an exact solution technique using a programming solver (CPLEX) with adequate preprocessing techniques. Second, we design a heuristic based on Variable Neighborhood Search (VNS) which iteratively improves the best-known solution through the enumeration and exploration of neighborhoods. Third, we develop a novel VNS guidance framework based on Deep Reinforcement Learning (DRL), where neighborhood operations are selected based on the policy obtained from reinforcement learning. These policies are learned ad-hoc for the to-be-solved problem at hand. We perform a rich set of computational experiments, which highlight the efficiency and effectiveness of the reinforcement learning-based solution technique.

While the primary focus of this study is related to the algorithmic approach to solving integrated airline recovery problems, it also provides multiple extensions on existing literature regarding data acquisition and model formulation, summarized below. First, we develop a set of compact algorithms to generate simulated airline operational datasets in an integrated manner, including flight schedule, crew pairing and passenger itinerary. Despite the relatively low resolution of these simulated datasets as compared with the existing literature on airline scheduling, these datasets are well-tailored for the airline disruption recovery model and subject to most of the regular constraints in airline operations. These algorithms can also generate datasets of various scales, which supports the scalability analysis of our proposed solution methodology. Second, we further introduce a set of realistic constraints regarding crew pairing process and passenger itinerary generation, as supplementary to the work of Arıkan et al. (2017). The introduced constraints related to crew duty include minimum/maximum sit-times between consecutive flights, maximum total flying time, maximum duty time (i.e., flying times and the sit-times in duty) and maximum number of landings. The introduced constraints related to passenger itinerary include minimum/maximum transfer time between consecutive flights and maximum number of connecting flights. The major contribution of this study is summarized as follows:

- We consider more practical constraints regarding aircraft/crew/passenger recovery in airline operations practice and manage to incorporate them into the state-of-the-art flight network-based model.
- We develop a set of compact algorithms to generate realistic datasets tailored to the integrated airline disruption recovery problem.
- We design a VNS-based heuristic which iteratively improves the best-known solution through the enumeration and exploration of neighborhoods.
- We develop a novel DRL-based VNS guidance framework which selects neighborhood operations based on the pre-trained policy, and we demonstrate its efficiency and accuracy through comparison with CPLEX results and VNS baseline results.

The remainder of this study is structured as follows. Section 2 reviews the literature concerning airline disruption recovery. Section 3 introduces the model formulation and provides the relevant intuition for the model through a synthesized example. Section 4 develops the three solution techniques compared in this study, including an exact algorithm and two heuristics. Section 5 reports on a wide range of computational experiments which explore the trade-off between solution optimality and computation time. Section 6 concludes this study and discusses a set of directions for future work.

2. Literature review

The modeling and computational techniques for airline disruption management have been rapidly developed in the literature. Previous efforts typically addressed irregular operations in a sequential manner, such that issues related to the aircraft fleet, crew, and passengers are resolved respectively (Barnhart, 2009; Xiong and Hansen, 2013; Wen et al., 2021). Section 2.1 first discusses research on the aircraft recovery problem, after which Section 2.2 reviews the airline recovery literature covering multiple resources, and Section 2.3 discusses the literature on the reinforcement learning approach for air transport management.

2.1. Aircraft recovery

As aircraft are the most constraining and expensive resource in aviation management, previous efforts mainly focused on aircraft recovery which can be formulated as follows: given a flight schedule and a set of disruptions, determine which flights to be cancelled or delayed, and re-assign the available aircraft to the flights such that the disruption cost is minimized (Hassan et al., 2021).

Teodorović and Guberinić (1984) were the first authors that discussed the minimization of passenger delays in the Aircraft Recovery Problem (ARP). The problem is formulated as a network flow problem in which flights are represented by nodes and arcs indicate time losses per flight. However, aircraft maintenance constraints are ignored and they assume that all itineraries contain only a single flight leg. They utilized the branch & bound method to solve the problem with a network of eight flights operated by three aircraft. This work was further extended in Teodorović and Stojković (1995) in which constraints including crew and aircraft maintenance were incorporated into the model. Thengvall et al. (2003) presented a multi-commodity flow model that considers flight delay, flight cancellation and through flights with flight copies and extra arcs. A tailored heuristic is adopted to solve the Lagrangian relaxation model. However, as the authors pointed out, a number of practical constraints cannot be embedded in the model due to their scenario and time-dependent nature.

In Eggenberg et al. (2010), a general modeling approach based on the time-band network was proposed. Each entity (i.e., an aircraft, a crew member or a passenger) is assigned with a specific recovery network and the model considers entity-specific constraints. The column generation algorithm is utilized to solve the majority of instances within 100 s, but the paper reports that the runtime exceeds one hour for the worst case, which may not meet the demand of airlines in seeking instant recovery solutions. Liang et al. (2018) incorporated exact delay decisions (instead of flight copies) based on airport slot resources constraints, which is similar to our approach concerning the modeling of recovery decisions. Aircraft maintenance swaps and several other recovery decisions are also implemented through the pricing problem of column generation. However, realistic factors such as maintenance capacity at airports are not considered in the proposed model.

2.2. Integrated recovery

In the recent decade, there has been a trend towards integrating more than one resource in recovery models. The interdependencies between aircraft, crew, and passengers cannot be fully captured by sequential optimization approaches, leading to sub-optimal recovery solutions. In this section, we discuss some of these integrated models. Table 1 summarizes existing literature on airline recovery problems.

Petersen et al. (2012) proposed an integrated optimization model in which four different stages are distinguished throughout the recovery horizon: schedule recovery, aircraft assignment, crew pairing, and passenger assignment. The schedule is repaired by flying, delaying or cancelling flights, and aircraft are assigned to the new schedule. Then, the crew is assigned to the aircraft rotations and the passenger recovery ensures that all passengers reach their final destination. Upon testing their model on data from a regional US carrier with 800 daily flights, the authors found that their integrated approach always guarantees a feasible solution. However, the runtime of the proposed solution ranges between 20–30 min, which may not meet the demand of airlines seeking efficient recovery solutions. Besides, the framework still works in a sequential manner rather than a fully integrated one, which may leads to sub-optimal recovery solutions.

The time–space network representation has been frequently adopted in the integrated recovery problem (Marla et al., 2017; Bratu and Barnhart, 2006; Lee et al., 2020), where the nodes in time–space networks are associated with airport and time, and an arc between two nodes indicates either a flight or a ground arc. Marla et al. (2017) incorporated flight planning (i.e., cruise speed control) to the set of traditional recovery decisions, which is also considered in our approach. Departure time decisions are modeled by creating copies of flight arcs, while the cruise speed decisions are modeled by creating a second set of flight copies for different cruise speed alternatives for each departure time alternative. This approach requires discretization of cruise speed options and greatly increases the size of generated networks, leading to computationally intractable issues when dealing with large-scale instances. Accordingly, the authors proposed an approximation model and tested their approach on data from a European airline with 250 daily flights, concluding that their approach reduces total costs and passenger-related delay costs for the airline.

In an attempt to avoid discretization of cruise speed options and reduce the size of generated networks, Arıkan et al. (2017) managed to formulate the recovery decisions (e.g., flight timing, aircraft rerouting, and cruise speed control) over a much smaller flight network with flight timing decisions in continuous settings. Since all entities (aircraft, crew members and passengers) are transported through a common flight network, the interdependencies among different entity types can be clearly captured. Based on Aktürk et al. (2014), the authors implemented aircraft cruise speed control and proposed a conic quadratic mixed-integer programming formulation. The model can evaluate the passenger delay costs more realistically since it explicitly models each individual passenger. The authors test the model on a network of a major U.S. airline and the optimality gap and runtime results

Table 1
Summary of existing literature on airline recovery problems

Reference	Network	Delay modeling	Multi-fleet	Maintenance	Cruise speed	Aircraft recovery	Crew recovery	Passenger recovery	Objective	Solution algorithm
Thengvall et al. (2003)	Time–space	Flight copies	✓			✓			Max revenue	Lagrangian relaxation
Eggenberg et al. (2010)	Time–space	Flight copies		✓		✓		✓	Min aircraft recovery and operation cost	Column generation
Petersen et al. (2012)	Time–space	Event-driven flight copies	✓	✓		✓	✓		Min schedule, aircraft, crew and passenger recovery cost	Benders decomposition, column generation
Hu et al. (2016)	Flight connection	Decide dynamically	✓			✓		✓	Min delay, reassignment and refund cost	Greedy randomized adaptive search procedure
Marzuoli et al. (2016)	Time–space	Decision variables				✓			Min passenger recommodation cost	Branch-and-bound
Sinclair et al. (2016)	Time–space	Flight copies	✓	✓		✓			Min aircraft recovery cost and passenger related cost	Large neighborhood search, column generation
Arıkan et al. (2017)	Flight connection	Decision variables	✓	✓	✓	✓	✓	✓	Min aircraft, crew recovery cost and passenger delay cost	Conic quadratic reformulation
Liang et al. (2018)	Flight connection	Adaptive delay		✓		✓			Min aircraft recovery cost	Column generation
Lee et al. (2020)	Time–space	Flight copies	✓			✓			Min expected recovery cost	Rolling algorithm
Cadarso and Vaze (2021)	Time–space	Decision variables	✓	✓	✓	✓		✓	Min operation, passenger and crew recovery cost	Branch-and-bound
Our paper	Flight connection	Decision variables	✓	✓	✓	✓	✓	✓	Min flight cancellation, passenger delay cost, etc.	DRL guided variable neighborhood search

show the feasibility of their approach. In this paper, we propose our integrated recovery model based on Arıkan et al. (2017), and we further introduce a set of realistic constraints regarding crew pairing process and passenger itinerary generation. In general, the approach greatly enhances efficiency and scalability as compared to other state-of-the-art studies.

In Hu et al. (2016), aircraft and passenger recovery decisions were integrated and a mixed-integer programming model was developed based on the passenger reassignment relationship. A randomized adaptive search algorithm was proposed accordingly to derive a recovery solution with reduced cost and passenger disruption. Cadarso and Vaze (2021) took cruise speed as a decision variable with a linear extra fuel cost in the objective function. Additionally, potential passenger losses are taken into account and related airfare revenue is optimized as well in the presented integrated model.

2.3. Reinforcement learning approach for air transport management

Machine learning methods are versatile tools to utilize and harness the hidden power of the data, and have been widely integrated in various domains of operations management (Herrema et al., 2019; Khan et al., 2021; Filom et al., 2022; Bongiovanni et al., 2022; Yan et al., 2022; Basso et al., 2022; Alcaraz et al., 2022). In recent years, there has been growing interest in the use of Reinforcement Learning (RL) for solving Air Transport Management (ATM) problems, which involves decision-making under uncertainty. RL algorithms can learn from past experiences and make decisions in real time, allowing them to quickly adapt to changing circumstances. Besides, RL algorithms can identify and exploit patterns in the data to make more informed decisions, leading to improved efficiency compared to rule-based or heuristic methods.

Ruan et al. (2021) used the RL-based Q-learning algorithm to solve the operational aircraft maintenance routing problem. Upon benchmarking with several meta-heuristics, they concluded that the proposed RL-based algorithm outperforms these meta-heuristics. Hu et al. (2021) proposed a RL-driven strategy for solving the long-term aircraft maintenance problem, which learned the maintenance decision policies for different aircraft maintenance scenarios. Experimental results on several simulated scenarios show that the RL-driven strategy performs well in adjusting its decision principle for handling the variations of mission reward. Kravaris et al. (2022) proposed a deep multi-agent RL method to handle demand and capacity imbalances in actual air traffic management settings with many agents, which addressed the challenges of scalability and complexity in the proposed problem. Pham et al. (2022) utilized the RL approach to solve the conflict resolution problem with surrounding traffic and uncertainty in air traffic management. The conflict resolution problem is formulated as a sequential decision-making problem and they used the deterministic policy gradient algorithm to deal with a complex and large action space.

To overcome the limitations of previous integrated recovery studies, we propose the RL approach in which we interpret the problem as a sequential decision-making process and improve the behavior of the agent by trial and error. Based on the aforementioned literature on applying RL to ATM problems, we summarized three main advantages to use RL for the integrated recovery problem. First, since RL is a typical agent-based model, the policy that the agent learned for a set of disruptions can be reused for other sets of disruptions. Reapplying the learned policy could accelerate the convergence process as compared to learning the policy from scratch (Šemrov et al., 2016; Lee et al., 2022), which is also in accordance with our experimental results. Second, since RL is a simulation-based method, it can handle complex assumptions in integrated recovery problem, where many realistic conditions and factors could affect airline operations (Lee et al., 2022). By incorporating these conditions in the simulation

(i.e., environment), RL could solve the integrated recovery problem under realistic situations. Third, RL is more flexible when designing the reward functions, which can meet various objectives than traditional operations research methods. In our study, we define the reward as the change of objective value after state transition. This function turns into a penalty if the next-state objective is higher than the current-state objective.

3. Model

Section 3.1 first provides an overview of the underlying model concepts. Section 3.2 introduces the actual model formulation. Section 3.3 highlights the properties of the model based on a synthesized example.

3.1. Overview

We illustrate the flight network representation for the airline disruption recovery problem based on Arıkan et al. (2017) first. Given an original schedule of the airline and a set of disruptions, the goal of the airline disruption recovery problem is to find the minimum-cost recovery actions by adapting the operations of aircraft, crew members and passengers within the recovery horizon $[T_0, T_1]$. Since all entity types (aircraft, crew member and passenger) are transported through a common flight network, the interdependencies among different entity types can be clearly defined. Parameters for the flight network representation are listed in Table 2, and essential concepts are explained below.

The original schedule of an airline serves as the input to the airline disruption recovery problem. Parameters related to scheduled flights including origin/destination airports, cruise time/flight time and lower/upper bounds of arrival/departure times are predetermined by the airline. The set of *scheduled flight nodes* F is defined as the set of scheduled flights within the recovery horizon. Throughout this paper, we use the term *entity* to refer to an aircraft, a crew member, or a passenger. Parameters related to the dynamic state of an entity include scheduled origin/destination airports, ordered set of flights within recovery horizon, minimum connection time and bounds of departure/arrival times (see Table 2).

The network contains four types of nodes, the source/sink node, must-visit-node (or must-node) and scheduled flight nodes, explained below.

- The *source node* for entity u representing the initial status of u is denoted by s^u, which satisfies $Des_{s^u} = Ori^u$, $EAT_{s^u} = EDT^u$, $MinCT^u_{s^u,f} = MaxCT^u_{s^u,f} = 0$ $(\forall f \in F)$, $D^u_{s^u} = -1$. Flow of entity u through a link between its source node and a flight node f indicates that the first flight assigned to u is f in the recovery. Connection times of these links do not belong to flight connections, thus are set to zero.
- The *sink node* for entity u representing the final status of u is denoted by t^u which satisfies $Ori_{t^u} = Des^u$, $LDT_{t^u} = LAT^u$, $MinCT^u_{g,t^u} = MaxCT^u_{g,t^u} = 0$ $(\forall g \in F)$, $D^u_{t^u} = +1$. Flow of entity u through a link between a flight node g and its sink node indicates that the last flight assigned to u is g in the recovery. Similarly, connection times of links between them are set to zero.
- The *must-nodes* are intended to model the restrictions of operating entities within the recovery horizon such as scheduled aircraft maintenance or scheduled rest periods of crew members. Must-nodes are not used for passengers. For each must-node m of entity u, $Ori_m = Des_m$ refers to the location of the activity. Besides, $MinCT^u_{fm} = MaxCT^u_{fm} = MinCT^u_{mg} = MaxCT^u_{mg} = 0$ $(f, g \in F)$, $D^u_m = 0$.
- The *scheduled flight nodes*. The demand of flight nodes are set to zero, i.e., $D^u_f = 0$ $(f \in F, u \in U)$.

A link (f, g) may indicate a flight connection (if $f, g \in F$), the beginning of the operation of an entity (if $f = s^u$), the end of the operation of an entity (if $g = t^u$), or connections with must-nodes (if f or $g \in M^u$). The *connecting criterion* for a node pair is as follows:

$$E = \{(f,g) : Des_f = Ori_g, LDT_g \geq EAT_f + MMCT_{fg},\ f, g \in \mathcal{V}\} \tag{1}$$

In addition, *external links* (i.e., $(f, g) \notin E$) are inserted into the network to incorporate recovery actions such as ferrying aircraft, deadheading crew members or reallocating passengers to other means of transportation.

The structure of the flight network is visualized in Fig. 1. The flight nodes which represent scheduled flights are shown in the middle, and the connections among them are generated according to Eq. (1). The nodes at the left side and right side of flight nodes are source nodes and sink nodes, respectively. For each entity, the links emanating from source nodes represent the possible choices of its first flight, while those directing to sink nodes represent the possible choices of its last flight. The must-nodes which are shown at the bottom represent restrictions to specific entities. Three external arcs are shown at the top of the network, which may correspond to different recovery actions for entities.

3.1.1. Disruption and recovery

All disruptions are modeled by modifying the parameters of entities and specific parts of the network. The four disruption types considered in our model are highlighted below.

- *Flight departure delay disruption* refers to the delays due to external reasons such as poor weather conditions, airport congestion or irregularities in ground operations. These disruptions are represented by updating SDT_f as $SDT_f + DD_f$, if flight f experiences a departure delay of DD_f in min.

Table 2
Parameters and sets for the flight network representation.

Parameters:	
T_0 (T_1)	The start (end) time of the recovery horizon
Ori_f (Des_f)	Origin (destination) airport of flight f
SDT_f (SAT_f)	Scheduled departure (arrival) time of flight f in the original schedule
LDT_f (LAT_f)	Latest allowable departure (arrival) time of flight f
ΔT_f^u	Cruise time compression limit of aircraft u for flight f
FT_f^u	Scheduled flight time of flight f when operated by aircraft u
EDT_f	Earliest possible time that flight f is allowed to depart, $EDT_f = SDT_f$
EAT_f	Earliest possible arrival time of flight f, $EAT_f = SDT_f + \min_{u\in U^{ac}}\{FT_f^u - \Delta T_f^u\}$
$MinCT_{fg}^u$	Minimum connection time (aircraft turnaround time, crew sit time, passenger transfer time) required for entity u between flights f and g
$MaxCT_{fg}^u$	Maximum allowable connection time for entity u between flights f and g
$MMCT_{fg}$	Minimum of minimum connection times among all entities ($u \in U$) between f and g
Ori^u	Location of entity u at the beginning of the recovery horizon (e.g., $Ori^u = Ori^u_{f_1}$)
Des^u	Planned destination of entity u at the end of the recovery horizon (e.g., $Des^u = Des_{f_n^u}$)
EDT^u	Earliest time that the first flight of entity u can depart (i.e., ready time), $EDT^u = \max\{T_0, SAT_{f_0^u} + MinCT^u_{f_0^u f_1^u}\}$ (the definitions of f_0^u and f_1^u are explained in the definition of $\mathcal{F}^u$)
LAT^u	Latest time that entity u needs to arrive at Des^u to catch up with its schedule, $LAT^u = \min\{T_1, SDT_{f_{n+1}^u} - MinCT^u_{f_n^u f_{n+1}^u}\}$
D_f^u	Demand of node f for entity u, $D_f^u \in \{-1, 0, 1\}$
MFT^u	Maximum total flying time (i.e., the total number of hours of actual flying time) for crew $u \in U^{cr}$
MDT^u	Maximum duty time (i.e., the flying times and the sit-times in the duty) for crew $u \in U^{cr}$
MNL^u	Maximum number of landings (connecting flights) for crew (passenger) $u \in U^{cr} \cup U^{ps}$
Sets:	
$\mathcal{T}$	The set of entity types including aircraft, crew member and passenger, $\mathcal{T} = \{ac, cr, ps\}$, $t \in \mathcal{T}$
U (U^t)	The set of entities (of type t) relevant to the problem, $U = \cup_{t\in\mathcal{T}} U^t$, $u \in U$
$\mathcal{F}$	The set of all scheduled flights of the airline
F	The set of flight nodes relevant to the problem, $F = \{f \in \mathcal{F} : SDT_f \geq T_0, EAT_f \leq T_1\}$
$\mathcal{F}^u$	Ordered set of scheduled flights originally assigned to entity u where flights with nonpositive subscripts are scheduled to be operated prior to T_0; flights with subscripts $n+1, n+2, \ldots$ are scheduled to be operated after T_1 ; and the flights with subscripts $1, \ldots, n$ are included in the recovery horizon $\mathcal{F}^u = \{\ldots, f_{-1}^u, f_0^u, f_1^u, \ldots, f_n^u, f_{n+1}^u\}$
F^u	Ordered set of flights originally assigned to entity u within the recovery horizon $F^u = \{f \in \mathcal{F}^u : SDT_f \geq T_0, EAT_f \leq T_1\} = \{f_1^u, f_2^u, \ldots, f_n^u\}$
M (M^u)	The set of must-nodes (of entity u), $M = \cup_{u\in U} M^u$, $m \in M$
$\mathcal{V}$	The set of nodes of the network, $\mathcal{V} = F \cup (\cup_{u\in U}\{s^u, t^u\}) \cup M$
$\mathcal{E}$	The set of links, $\mathcal{E} = E \cup \Gamma$
E	The set of links generated according to Eq. (1)
Γ (Γ^u)	The set of external links (of entity u), $\Gamma = \cup_u \Gamma^u$

- *Flight cancellation disruption* refers to the cancellations by external sources such as poor weather conditions or aircraft mechanical problems. Let $\mathcal{D}^{[fc]}$ be the set of cancelled flights, then all the nodes in $\mathcal{D}^{[fc]}$ together with all links emanating from or incoming to these nodes are removed.
- *Delayed ready time disruption* refers to the lag of ready time (i.e., EDT^u) of particular entities, for instance, unscheduled maintenance of aircraft or late arrivals of crew members. These disruptions are modeled by updating EDT^u as $EDT^u + RD^u$ if entity u experiences a ready time delay of RD^u in minutes.
- *Airport closure disruption* refers to the cancellation of all departures and arrivals for a certain period by the airport, due to external reasons such as poor weather conditions. Let $\mathcal{D}^{[ac]}$ be the set of closed airports and $a \in \mathcal{D}^{[ac]}$ be an airport experiencing

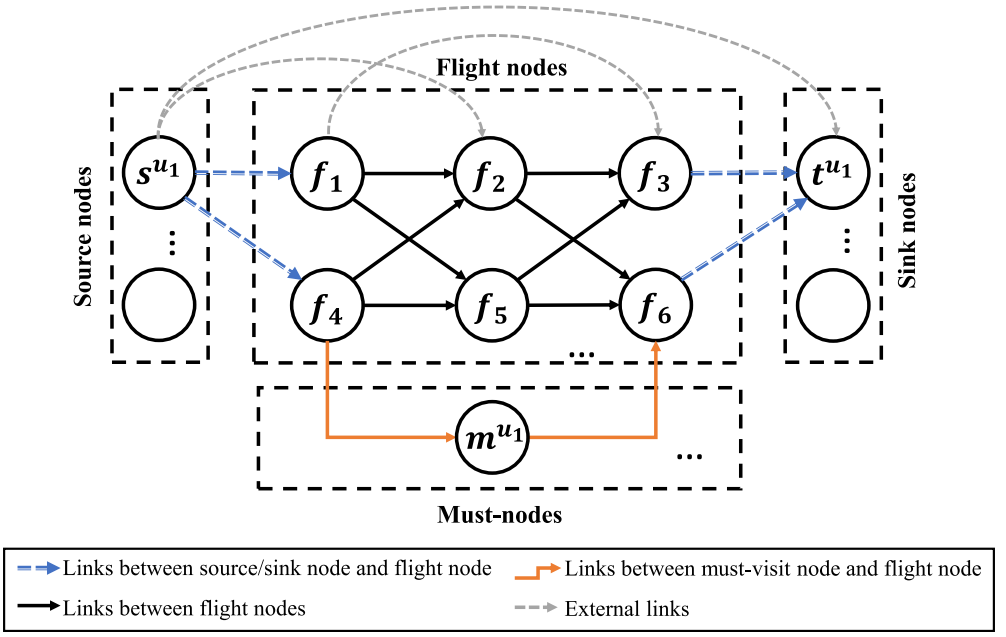

Fig. 1. The structure of flight network $G = (\mathcal{V}, \mathcal{E})$. The flight nodes which represent scheduled flights are shown in the middle, and the connections among them are generated according to Eq. (1). The nodes at the left side and right side of flight nodes are source nodes and sink nodes, respectively. The must-nodes are shown at the bottom. Three external arcs are shown at the top of the network.

a closure during $[ST_a, ET_a]$. This disruption may lead to two kinds of consequence. First, some flights may be cancelled during this closure, and they are inserted to the set of cancelled flights. Second, some other flights may avoid the closure period and still be operated by rescheduling the departure/arrival times or altering the cruise speed. Mathematically, for each airport $a \in D^{[ac]}$,

- If $Ori_f = a$, $SDT_f > ST_a$ and $LDT_f < ET_a$, then $D^{[fc]} = D^{[fc]} \cup f$
- If $Des_f = a$, $EAT_f > ST_a$ and $LAT_f < ET_a$, then $D^{[fc]} = D^{[fc]} \cup f$
- If $Ori_f = a$, $SDT_f < ST_a$ and $LDT_f > ST_a$, then $LDT_f = ST_a$
- If $Ori_f = a$, $SDT_f < ET_a$ and $LDT_f > ET_a$, then $SDT_f = ET_a$
- If $Des_f = a$, $EAT_f < ST_a$ and $LAT_f > ST_a$, then $LAT_f = ST_a$
- If $Des_f = a$, $EAT_f < ET_a$ and $LAT_f > ET_a$, then $EAT_f = ET_a$

Based on the flight network representation, the goal of the airline disruption recovery problem is to *find the minimum-cost recovery plan for each flight and the flow of entities from their source nodes to sink nodes provided that must-nodes are visited*. The optimal recovery plan for flights and flow of entities correspond to the optimal recovery decision over a search space, which includes the following recovery actions: departure holding, flight cancellation, cruise speed control, aircraft and crew swapping, aircraft and crew rerouting and passenger reaccommodation.

3.2. Mathematical formulation

In this section, we present the formal mathematical formulation. The decision variables are reported in Table 3. The objective function aims to minimize the additional operational cost of the airline due to disruptions. The five terms of the objective function (from left to right) refer to flight cancellation cost, passenger delay cost, external link cost, additional fuel cost and following schedule cost, respectively. The additional fuel cost is the consequence of greater fuel consumption as a result of increased cruise speed. Since entities may be rerouted without leading to flight cancellation or arrival delay, it is desirable for entities to use their original routing unless the rerouting options can help mitigate disruptions; leading to the small negative following schedule cost. The mathematical formulation is as follows:

$$\min \quad \sum_{f\in F} C_f^{[c]}\kappa_f + \sum_{u\in U^{ps}} C_u^{[pd]} delay_u + \sum_{u\in U}\sum_{e\in \Gamma^u} C_e^{[e]}\beta_e^u + \sum_{f\in F} C^{[f]}\left(\sum_{u\in U^{ac}} fc_{fu} - FC_f\right) + \sum_{t\in T}\sum_{u\in U^t}\sum_{(f,g)\in SE^u} C_t^{[fs]}\beta_{fg}^u \tag{2}$$

Table 3
Decision variables, sets and parameters of the mathematical model.

Decision	Variables:
β_{fg}^{u}	(Binary) equals to 1 if entity u flows through link (f,g), and 0 otherwise
κ_f	(Binary) equals to 1 if flight f is cancelled, and 0 otherwise
α_{fu}	(Binary) equals to 1 if flight f is assigned by aircraft $u \in U^{ac}$
dt_f	(Continuous) the actual departure time of flight f, $dt_f \in [EDT_f, LDT_f]$
at_f	(Continuous) the actual arrival time of flight f, $at_f \in [EAT_f, LAT_f]$
δt_f	(Continuous) the amount of cruise time compression of flight f, $\delta t_f \geq 0$
v_{fu}	(Continuous) the cruise speed of aircraft u for flight f if f is assigned to u, $v_{fu} \in [V_{fu}, \overline{V}_{fu}]$, and 0 otherwise
$fc_{fu}(v_{fu})$	(Continuous) the fuel consumption (kg) of flight f if it is operated by aircraft u at cruise speed v_{fu} (km/min) in the recovered schedule, and 0 otherwise
$delay_u$	(Continuous) The arrival delay of the passenger $u \in U^{ps}$ (min)
Sets:	
$\mathcal{T}^{OP}$	The set of operating entity types including aircraft and crew member, i.e., $\mathcal{T}^{OP} = \{ac, cr\}$
DC^u	The set of departure-critical links, $DC^u = \{(s^u, g \in E : EDT_g < EDT^u)\}$
AC^u	The set of arrival-critical links, $AC^u = \{(f, t^u) \in E : LAT_f > LAT^u\}$
CC^u	The set of connection-critical links, $CC^u = \{(f,g) \in \mathcal{E} : f, g \in F \cup M\}$
SE^u	The set of scheduled links of entity u, obtained by the flight schedule or scheduled itinerary
$U^{ac}(u)$	The set of aircraft that crew member $u \in U^{cr}$ is eligible to operate
Parameters:	
Req_f^t	The number of entities of type t needed to operate flight f
Cap^r	The seat capacity of aircraft $r \in R^{ac}$
$C^{[e]}$	The cost of unit flow on external link e
$C_f^{[c]}$	The flight cancellation cost of flight f
$C^{[f]}$	The jet fule price per kg
$C_u^{[pd]}$	The per minute delay cost of passenger u
$C_t^{[fs]}$	The negative following schedule cost coefficient of entity type $t \in \mathcal{T}$
V_{fu}, $\overline{V}_{fu}$	The scheduled (maximum) cruise speed of aircraft u for flight f
Dis_f	The distance flown at the cruise stage of flight f
Λ_{ui}	The aircraft-specific fuel consumption coefficients, $i = 1, 2, 3, 4$
FC_f	The scheduled fuel consumption of flight f, i.e., $FC_f = fc_{fu}(V_{fu})$

s.t.

$$\sum_{f:(f,g)\in\mathcal{E}} \beta_{fg}^{u} - \sum_{h:(g,h)\in\mathcal{E}} \beta_{gh}^{u} = D_g^u \qquad u \in U, g \in \mathcal{V} \tag{3}$$

$$\sum_{u\in U^t} \left(\sum_{g:(f,g)\in\mathcal{E}} \beta_{fg}^{u} \right) = \left(1 - \kappa_f\right) Req_f^t \qquad t \in \mathcal{T}^{OP}, f \in F \tag{4}$$

$$\sum_{g:(f,g)\in\mathcal{E}} \beta_{fg}^{u} \leq \left(1 - \kappa_f\right) \qquad t \in \mathcal{T} \backslash \mathcal{T}^{OP}, u \in U^t, f \in F \tag{5}$$

$$\sum_{u\in U^{ps}} \sum_{g:(f,g)\in\mathcal{E}} \beta_{fg}^{u} \leq \sum_{u\in U^{ac}} \sum_{g:(f,g)\in\mathcal{E}} \beta_{fg}^{u} Cap^u \qquad f \in F \tag{6}$$

$$\beta_{fg}^{u} \leq \sum_{a\in U^{ac}(u)} \beta_{fg}^{a} \qquad (f,g) \in \mathcal{E}, u \in U^{cr} \tag{7}$$

$$\sum_{g:(m,g)\in\mathcal{E}} \beta^u_{mg} = 1 \qquad u \in U, m \in M^u \tag{8}$$

$$at_f = dt_f + \sum_{u\in U^{ac}} \left(\sum_{g:(f,g)\in\mathcal{E}} \beta^u_{fg} \right) FT^u_f - \delta t_f \qquad f \in F \tag{9}$$

$$dt_g \geq EDT_u \beta^u_{s^u g} \qquad u \in U, (s^u, g) \in DC^u \tag{10}$$

$$at_f \leq LAT_f + \left[LAT^u - LAT_f \right] \beta^u_{f t^u} \qquad u \in U, (f, t^u) \in AC^u \tag{11}$$

(cont.)

$$dt_g \geq at_f + MinCT^u_{fg} \beta^u_{fg} - LAT_f \left(1 - \beta^u_{fg}\right) \qquad u \in U, (f,g) \in CC^u \tag{12}$$

$$dt_g \leq at_f + MaxCT^u_{fg} \beta^u_{fg} + LDT_f \left(1 - \beta^u_{fg}\right) \qquad u \in U, (f,g) \in CC^u \tag{13}$$

$$\sum_{f\in F} \left(\sum_{g:(f,g)\in\mathcal{E}} \beta^u_{fg} \right) \left(at_f - dt_f \right) \leq MFT^u \qquad u \in U^{cr} \tag{14}$$

$$\left(\sum_{(f,t^u)\in\mathcal{E}} at_f \beta^u_{f t^u} \right) - \left(\sum_{(s^u,g)\in\mathcal{E}} dt_g \beta^u_{s^u g} \right) \leq MDT^u \qquad u \in U^{cr} \tag{15}$$

$$\sum_{(f,g)\in\mathcal{E}} \beta^u_{fg} - 1 \leq MNL^u \qquad u \in U^{cr} \cup U^{ps} \tag{16}$$

$$\alpha_{fu} = \sum_{g:(f,g)\in\mathcal{E}} \beta^u_{fg} \qquad f \in F, u \in U^{ac} \tag{17}$$

$$\alpha_{fu} V_{fu} \leq v_{fu} \leq \alpha_{fu} \overline{V}_{fu} \qquad f \in F, u \in U^{ac} \tag{18}$$

$$\delta t_f = Dis_f / V_{fu} - \sum_{u\in U^{ac}} \alpha_{fu} \left(Dis_f / v_{fu} \right) \qquad f \in F \tag{19}$$

$$\delta t_f \leq \sum_{u\in U^{ac}} \sum_{g:(f,g)\in\mathcal{E}} \beta^u_{fg} \Delta T^u_f \qquad f \in F \tag{20}$$

$$fc_{fu} = \alpha_{fu} Dis_f \left(\Lambda_{u1} v^2_{fu} + \Lambda_{u2} v_{fu} + \frac{\Lambda_{u3}}{v^2_{fu}} + \frac{\Lambda_{u4}}{v^3_{fu}} \right) \qquad f \in F, u \in U^{ac} \tag{21}$$

$$delay_u \geq at_f - SAT^u - \left(LAT_f - SAT^u \right) \left(1 - \beta^u_{f t^u}\right) \qquad u \in U^{ps}, f \in F, (f, t^u) \in \mathcal{E} \tag{22}$$

Constraints (3) are the flow balance constraints regarding each node in the network. Constraints (4) and (5) reveal the dependency between flights and operating entities, i.e., aircraft and crew members. If the required number of operating entities does not flow through the corresponding flight node, then this flight will be cancelled. Meanwhile, the other entities (i.e., passengers) cannot flow through this cancelled flight node as well. Constraints (6) are the aircraft seat capacity constraints and limit the passenger flow of each flight by the seat capacity of the assigned aircraft. Constraints (7) ensure that the crew members and aircraft assigned to each flight are compatible. Constraints (8) which involve the flow over must-nodes are associated with entity-specific restrictions. Constraints (9) are the flight time constraints which reveal the relationship between actual departure/arrival time and cruise speed compression. Constraints (10), which are imposed on a set of departure-critical links, ensure that the entity is ready at the departure time of its first flight. Likewise, Constraints (11) imposed on a set of arrival-critical links ensure that the arrival time of the last flight of an entity cannot be later than the latest arrival time of the entity. Constraints (12) and (13), which deal with a set of connection-critical links, ensure that the connection time between the minimum threshold and the maximum threshold should be provided between a pair of connecting flights with positive flow of entities. Constraints (14), (15) and (16) ensure that the total flying time, duty time and number of landings for each crew member do not exceed the corresponding maximum thresholds in the recovery horizon, respectively. Constraints (16) also restrict the number of connecting flights taken by the passengers. Constraints (17) to (21) present expressions for calculating the cruise time compression and the fuel consumption of flights to model the trade-off between disruption cost and extra fuel cost due to increased cruise speed (Arıkan, 2014). Constraints (22) consider the realized arrival times of passengers to evaluate exact passenger delay linearly.

3.3. Synthesized example with exemplified disruptions

In this section, we illustrate the recovery problem with a synthesized example, which mainly consists of three aircraft and 13 flights. The original flight schedule of an airline within the recovery horizon (from 5 AM to 2 AM the next day) is presented in Table 4. Three aircraft with their seat capacity and six crew teams are involved in this example. Each flight is assumed to be operated by a crew team in this example. All the operating entities (i.e., aircraft and crew teams) are assumed to be located at the origin airport of their first flight at 5 AM. The minimum required connection time for all types of entities is set to 30 min (Barnhart et al., 2014; Wang and Jacquillat, 2020), while the latest departure (arrival) time of flights is set to two hours after the scheduled departure

Table 4
The original flight schedule of an airline. The abbreviations Cap, Ori, Des, SDT, SAT, CrsTime, Dist and Pax refer to seat capacity, origin airport, destination airport, scheduled departure time of flight, scheduled arrival time of flight, scheduled cruised time, distance (km) and number of passengers, respectively. All the departure and arrival times in the table are converted to the local time at airport LAX.

Tail (Cap)	Flight	Crew	Ori	Des	SDT	SAT	CrsTime	Dist	Pax
T00 (120)	F00	C00	LAX	ORD	05:38	09:08	03:00	2802	79
	F01	C00	ORD	ATL	11:39	12:52	00:43	976	91
	F02	C01	ATL	ORD	17:17	18:31	00:43	976	90
	F03	C01	ORD	DFW	21:51	23:28	01:06	1290	102
	F04	C01	DFW	ATL	(+1) 00:17	(+1) 01:45	00:58	1174	87
T01 (120)	F05	C02	ATL	ORD	05:48	07:01	00:43	976	83
	F06	C02	ORD	LAX	07:56	11:26	03:00	2802	84
	F07	C03	LAX	ORD	12:55	16:25	03:00	2802	80
	F08	C03	ORD	LAX	20:36	(+1) 00:06	03:00	2802	87
T02 (160)	F09	C04	LAX	ORD	05:47	09:17	03:00	2802	125
	F10	C04	ORD	DFW	11:39	13:16	01:06	1290	122
	F11	C05	DFW	ATL	16:20	17:48	00:58	1174	124
	F12	C05	ATL	DFW	22:44	(+1) 00:12	00:58	1174	138

Table 5
The original itinerary schedule of an airline. The abbreviations Itin, Ori, Des, SDT, SAT and Pax refer to passenger itinerary, origin airport, destination airport, scheduled departure time of itinerary, scheduled arrival time of itinerary and number of passengers, respectively. Both direct itineraries and two-hop itineraries are considered.

Itin	Flights	Ori	Des	SDT	SAT	Pax
I00	F00	LAX	ORD	05:38	09:08	52
I01	F00–F01	LAX	ATL	05:38	12:52	27
I02	F01	ORD	ATL	11:39	12:52	64
I03	F02	ATL	ORD	17:17	18:31	90
I04	F03	ORD	DFW	21:51	23:28	64
I05	F03–F04	ORD	ATL	21:51	(+1) 01:45	38
I06	F04	DFW	ATL	(+1) 00:17	(+1) 01:45	49
I07	F05	ATL	ORD	05:48	07:01	83
I08	F06	ORD	LAX	07:56	11:26	84
I09	F07	LAX	ORD	12:55	16:25	80
I10	F08	ORD	LAX	20:36	(+1) 00:06	87
I11	F09	LAX	ORD	05:47	09:17	125
I12	F10	ORD	DFW	11:39	13:16	76
I13	F10–F11	ORD	ATL	11:39	17:48	46
I14	F11	DFW	ATL	16:20	17:48	78
I15	F12	ATL	DFW	22:44	(+1) 00:12	138

Table 6
The recovery flight plan of the example. The abbreviations Cap, Ori, Des, RDT, RAT, CrsTime, Dist, Pax refer to seat capacity, origin airport, destination airport, recovery departure time of flight, recovery arrival time of flight, recovery cruise time, distance (km) and number of passengers, respectively.

Tail (Cap)	Flight	Crew	Ori	Des	RDT	RAT	CrsTime	Dist	Pax
T00 (120)	F09	C00	LAX	ORD	08:47	12:17	03:00	2802	44
	F01	C00	ORD	ATL	12:47	14:00	00:43	976	91
	F02	C01	ATL	ORD	17:17	18:31	00:43	976	90
	F03	C01	ORD	DFW	21:51	23:28	01:06	1290	102
	F04	C01	DFW	ATL	(+1) 00:17	(+1) 01:45	00:58	1174	87
T01 (120)	F05	C02	ATL	ORD	07:48	09:01	00:43	976	83
	F06	C02	ORD	LAX	09:31	13:01	03:00	2802	84
	F07	C03	LAX	ORD	13:31	17:01	03:00	2802	80
	F08	C03	ORD	LAX	20:36	(+1) 00:06	03:00	2802	87
T02 (160)	F00	C04	LAX	ORD	05:47	09:17	03:00	2802	160
	F10	C04	ORD	DFW	11:39	13:16	01:06	1290	122
	F11	C05	DFW	ATL	16:20	17:48	00:58	1174	124
	F12	C05	ATL	DFW	22:44	(+1) 00:12	00:58	1174	138

(arrival) time. The original itinerary schedule is tabulated in Table 5. Both direct itinerary and two-hop itinerary are considered. The disruption scenario is assumed to be a combination of a two-hour departure delay of F05 and a three-hour departure delay of F09; changing the departure time of F05 and F09 to 07:48 and 08:47, respectively. For the sake of simplicity, the must-nodes and external links are not considered in this example.

In the absence of recovery actions, the severe departure delay disruptions of F05 and F09 will propagate through the downstream flights of aircraft T01 and T02, leading to high passenger delay cost. The optimal solution (recovery plan) of this example is obtained by solving the MINP model through a commercial solver CPLEX. The recovery flight plan and recovery itinerary plan are tabulated in Tables 6 and 7. A swap action between tail T00 and T02 (i.e., T00 takes over F09 while T02 takes over F00) takes place as a result

Table 7
The recovery itinerary plan of the example. The abbreviations Skd Itin, Rec Itin, Skd Flt(s), Rec Flt(s) and Pax refer to schedule passenger itinerary, recovery passenger itinerary, flight legs in schedule itinerary, flight legs in recovery itinerary and number of passengers. Delay refers to the arrival delay of individual passengers and cost refers to the passenger delay cost.

Skd Itin	Rec Itin	Skd Flt(s)	Rec Flt(s)	Pax	Delay	Cost
I00	I00	F00	F00	52	00:09	479.3
I01	I16	F00–F01	F09–F01	27	01:07	1868.9
I02	I02	F01	F01	64	01:07	4430.0
I03	I03	F02	F02	90		
I04	I04	F03	F03	64		
I05	I05	F03–F04	F03–F04	38		
I06	I06	F04	F04	49		
I07	I07	F05	F05	83	02:00	10 201.0
I08	I08	F06	F06	84	01:34	8125.8
I09	I09	F07	F07	80	00:35	2912.8
I10	I10	F08	F08	87		
I11	I00	F09	F00	108		
I11	I11	F09	F09	17	03:00	3134.1
I12	I12	F10	F10	76		
I13	I13	F10–F11	F10–F11	46		
I14	I14	F11	F11	78		
I15	I15	F12	F12	138		

of the disruption on F09. Moreover, a crew swap of C00 and C04 also takes place between F09 and F00, otherwise, C04 will face overlapped flights F09 and F10 (the RAT of F09 is 12:17, which is later than the RDT of F10, 11:39). As for the recovery itinerary, a large proportion of passengers of I11 are rerouted to I00, since I11 is the itinerary with disrupted flight F09. Furthermore, a new itinerary I16 is created to accommodate the passengers of I01 regarding the first flight leg, since the original first flight leg F00 is fully seated in the recovery itinerary. For each scheduled itinerary, the actual passenger delay $RAT_f - SAT_g$ is computed, where f refers to the last flight of the recovery itinerary and g refers to the last flight of the scheduled itinerary. The delay cost is proportional to the passenger delay, with the coefficient $C_u^{[pd]}$ set to \$1.0242 (Ball et al., 2010a). The decreasing delays of I07, I08 and I09 indicate the *delay absorption* based on the adjustment of connecting times and cruise times. We provide more relevant information and draw more insights from the recovery solution of this example in Appendix A.

4. Solution methodology

In this section, three solution methods to solve our mixed-integer non-linear programming (MINP) model are developed. In Section 4.1, we present an exact method using the commercial solver CPLEX incorporated with preprocessing techniques following (Arıkan et al., 2017). Section 4.2 introduces our VNS-based solution framework, which explores solutions by the enumeration of well-defined neighborhoods. Section 4.3 explores how a deep learning-based guidance framework can improve the solutions from VNS significantly.

4.1. Exact solution using CPLEX incorporated with preprocessing techniques

A commercial mathematical programming solver, such as CPLEX, can be used to obtain an exact (or close-to-optimal) out-of-the-box solution to the MINP model. However, in order to obtain results for medium-sized model instances, a few modifications are necessary. Based on Arıkan et al. (2017), we further improve the model regarding the reformulation of nonlinearity, the preprocessing of the flight network and the aggregation of entities, explained briefly below.

- The MINP model can be reformulated as a *conic quadratic mixed-integer programming* (CQMIP) problem (Arıkan, 2014). Upon reformulation, the modified model features a linear objective function, as well as linear and conic quadratic constraints, which can be handled by commercial CQMIP solvers.
- Based on a recursive structure, the *Partial Network Generation Algorithm* (PNGA) efficiently generates the partial network of an entity, which covers all feasible paths for the entity to reach its destination and excludes any unvisited flight nodes.
- Due to the large number of individual entities (aircraft, crew members and passengers) in real-world scenarios, the *Aggregation Rule* forces that individuals (especially passengers) of the same partial network can be aggregated through simple modifications to the model without sacrificing optimality. The aggregated individuals can still be transported through different paths.

4.2. VNS-based solution framework

Given the high intrinsic complexity, solving large-scale airline disruption recovery problems by a commercial solver is inefficient even in the presence of the above modifications. To address this problem, we develop a VNS-based heuristic which iteratively improves the solution quality through the exploration of neighborhoods. Given the large solution space of the airline disruption recovery problem, our proposed VNS-based heuristic seeks to explore high-quality solutions by modifying and repairing the

Algorithm 1 VNS-based algorithm

Input: Schedule aircraft/crew team assignment (P_1, Q_1), # of neighborhoods $\mathcal{N}$, length of trajectory $\mathcal{L}_T$, baseline method $\mathcal{B}$
Output: Final objective value obj_1 and final incumbent recovery solution sol_1

$obj_1, sol_1 \leftarrow$ EVALUATOR(P_1, Q_1)
for each step $t \in \{1, ..., \mathcal{L}_T\}$ **do**
  $k \leftarrow 1$
  while $k \leq \mathcal{N}$ **do**
    $P_1dist, Q_1dist \leftarrow$ FLIGHTSTRINGDIST$(P_1, Q_1, \mathcal{B})$ // see Section 4.2.3
    $(P_2, Q_2) \leftarrow$ EXPLORENEIGHBORK$(k, P_1, Q_1, P_1dist, Q_1dist)$
    $obj_2, sol_2 \leftarrow$ EVALUATOR(P_2, Q_2) // see Section 4.2.1
    if $obj_2 \geq obj_1$ **then**
      $k \leftarrow k + 1$
    else
      $obj_1, sol_1, P_1, Q_1 \leftarrow obj_2, sol_2, P_2, Q_2$
    end if
  end while
end for
return obj_1, sol_1

Algorithm 2 Explore neighborhoods

function EXPLORENEIGHBORK$(k, P_1, Q_1, P_1dist, Q_1dist)$
  for each $X \in \{P, Q\}$ **do**
    $x_1^1, x_1^2 \leftarrow$ Sample a pair of flight strings in X_1 based on X_1dist
    $x_2^1, x_2^2 \leftarrow$ NEIGHBOROPERATION(k, x_1^1, x_1^2) // see Section 4.2.2
    $X_2 \leftarrow$ Override x_1^1, x_1^2 of X_1 by x_2^1, x_2^2
  end for
end function
return (P_2, Q_2)

scheduled assignment of flights to specific aircraft or crew teams, instead of fully searching for decision values on aircraft/crew routing, flight timing and passenger assignment. For the sake of simplicity, we assume $u \in U^{cr}$ refers to a crew team; the methodology can also be extended to individual crew members in a similar manner.

Unlike the flight network representation presented in Section 3, our VNS-based heuristic utilizes flight string representations (Barnhart et al., 1998) to define the neighborhood structures. A *flight string* for aircraft u, $p^u = \{f_1^u, \dots, f_k^u\}$ $(\forall u \in U^{ac})$ is a sequence of flights which can be operated by aircraft u successively and satisfies the connection criterion Eq. (1). A possible *aircraft assignment* is a combination of flight strings, i.e., $P = \{p^{u_1}, p^{u_2}, \dots, p^{u_{|U^{ac}|}}\}$. Note that $p^{u_i} \cap p^{u_j} = \emptyset$ $(\forall u_i, u_j \in U^{ac}, u_i \neq u_j)$ and $\cup_{u_i \in U^{ac}} p^{u_i} \subseteq F$. In a like manner, the flight string for crew team is denoted by $q^u = \{f_1^u, \dots, f_k^u\}$ $(\forall u \in U^{cr})$ and a possible *crew assignment* is denoted by $Q = \{q^{u_1}, q^{u_2}, \dots, q^{u_{|U^{cr}|}}\}$. $q^{u_i} \cap q^{u_j} = \emptyset$ $(\forall u_i, u_j \in U^{cr}, u_i \neq u_j)$ and $\cup_{u_i \in U^{cr}} q^{u_i} \subseteq F$. We denote the combination (P, Q) as a *state*.

An *evaluator* which evaluates a state approximates the objective function encompassing delay cost, cancellation cost and following schedule cost. Based on the configuration of assignments P and Q, the evaluator finally returns an objective value obj and a recovery solution sol, which consists of aircraft/crew team/passenger assignments and the flight timing.

The VNS-based heuristic incorporates the basic feature of sequential VNS where several neighborhoods of various structures are explored sequentially. Our heuristic is formalized in Algorithm 1 and 2. Initially, we extract the scheduled aircraft/crew team assignment from the original schedule to obtain an initial state. Then, a sequential VNS is applied to generate and explore different neighborhoods of the current state. Probability distributions over flight strings of aircraft/crew team assignment are generated based on various baseline methods, which can inform the sampling of flight string pairs during exploration in line 3 of Algorithm 2. Upon sampling, neighborhood operation k is performed on the flight string pairs, and thus a modified state (P_2, Q_2) can be generated and passed to the evaluator. If an improvement of objective value is found, we update the objective value, recovery solution, and the incumbent state respectively, otherwise the k is incremented for the next neighborhood. A local optimal solution is obtained when all neighborhood structures are explored without improvement to the objective value. In the following, we explain our approach in three aspects, the evaluator, neighborhood structure and distribution generator.

4.2.1. Evaluator

Given a state (i.e., the aircraft/crew team assignment (P, Q)), the goal of the evaluator is to determine the corresponding recovery solution in order to minimize the additional operational cost due to disruptions, including passenger delay cost, flight cancellation cost and following schedule cost. Note that the passenger delay is referring to the arrival delay of each individual passenger, i.e., the

difference between the actual arrival time and the scheduled arrival time of the passenger. The evaluator consists of three steps, as explained below.

1. *Determine the greedy flight timings given connection constraint*. In order to minimize the propagated delay experienced by the downstream flights, the departure/arrival times of flights are set as early as possible. For each flight string p^u in P, the departure time of flight f_i^u is $dt(f_i^u) = \max\{SDT(f_i^u), at(f_{i-1}^u) + MinCT^u\}$ if $i \neq 0$, otherwise $SDT(f_0^u)$. In addition, for crew teams and passengers with multiple legs of flights, the connection times between these consecutive flights should also be guaranteed to stay within the range defined by the minimum and maximum thresholds. Thus, we perform feasibility checks on these critical connections and adapt the corresponding departure/arrival time of flights in case of overlapped flight pairs or misconnected flight pairs. We also perform feasibility checks on crew duty constraints including maximum flying time, maximum duty time and maximum number of landings. The current state is rejected if one of the aforementioned checks is failed even after adaption.
2. *Reallocate passengers from disrupted itineraries to replaceable flights in a way to mitigate delay cost*. First, rank passenger itineraries by decreasing experienced delay based on the greedy flight timings. For each delayed passenger itinerary I_d, find the replaceable flights f_r in P with the same OD prefix as I_d and an earlier arrival time compared to the current arrival time of I_d, i.e., $at(f_r) < at(I_d)$. Next, for each replaceable flight f_r, the number of passengers reallocated from I_d to f_r (denoted as *rerouting group*) is $\Delta pax = \min\{remain(f_r), pax(I_d)\}$, where *remain* and *pax* refer to the number of remaining seats and passengers respectively. The number of passengers who belong to I_d but do not reroute (denoted as *staying group*) is $pax(I_d) - \Delta pax$. Then, three specific cases are discussed, respectively:

   - If the departure time of f_r is later than $SDT(I_d)$, which indicates rerouting passengers to a later flight, then the delay cost specific to I_d and f_r contains two components, for the rerouting group from I_d and the staying group from I_d, respectively,

$$\begin{aligned} Cost(I_d, f_r) = {} & C_u^{[pd]} \cdot \Delta pax \cdot \max\{at(f_r) - SAT(I_d), 0\} \\ & + C_u^{[pd]} \cdot [pax(I_d) - \Delta pax] \cdot \max\{at(I_d) - SAT(I_d), 0\} \end{aligned}$$

   - If the departure time of f_r is earlier than $SDT(I_d)$, an intentional *departure holding* for f_r is considered in order to accommodate passengers from I_d, i.e., $dt(f_r)$ is set to $SDT(I_d)$ and $at(f_r)$ is set to $SDT(I_d) + FT(f_r)$. In this case, the delay cost specific to I_d and f_r contains three components, for the rerouting group from I_d, the staying group from I_d, and the passenger from f_r, respectively,

$$\begin{aligned} Cost(I_d, f_r) = {} & C_u^{[pd]} \cdot \Delta pax \cdot \max\{at(f_r) - SAT(I_d), 0\} \\ & + C_u^{[pd]} \cdot [pax(I_d) - \Delta pax] \cdot \max\{at(I_d) - SAT(I_d), 0\} \\ & + C_u^{[pd]} \cdot pax(f_r) \cdot \max\{at(f_r) - SAT(f_r), 0\} \end{aligned}$$

   - This delayed itinerary I_d may also keep all its passengers, then the delay cost specific to I_d is only referred to the staying group from I_d,

$$Cost(I_d) = C_u^{[pd]} \cdot pax(I_d) \cdot \max\{at(I_d) - SAT(I_d), 0\}$$

   The best alternative (reallocating passengers to a specific f_r or keeping all passengers) is the one with minimum related delay cost. We repeat the same procedure for all delayed passenger itineraries.
3. *Compute the aforementioned cost terms based on flight timings and passenger allocation.* The passenger delay cost, flight cancellation cost and following schedule cost are computed according to Eq. (2).

4.2.2. Neighborhood operator

VNS unfolds its full advantage, concerning the ability to escape local minima, by exploring well-designed neighborhoods and corresponding systematic changes (Xu et al., 2021). In the following, we introduce four types of neighborhood structures to modify current state: *swap*, *cut*, *insert* and *delete*. Let $con(f_i, f_j)$ be a statement that f_i can be connected to f_j according to Eq. (1). Each time when line 3 of Algorithm 2 is called, a pair of flight strings $p^u = \{f_1^u, \dots, f_n^u\}$, $p^v = \{f_1^v, \dots, f_m^v\}$ $(u, v \in U^{ac})$ are randomly selected from P. A *swap* focuses on swapping two flight elements, while a *cut* and a *insert* can deal with flight substrings. A *delete* can be used to cancel flights. For convenience, let f_0^u and f_{n+1}^u (or f_{m+1}^u) denote the source node s^u and sink node t^u, respectively.

1. *Swap*: Given $p^u = \{f_1^u, \dots, f_{i-1}^u, f_i^u, f_{i+1}^u, \dots, f_n^u\}$ $(i = 1, \dots, n)$, then seek $p^v = \{f_1^v, \dots, f_{j-1}^v, f_j^v, f_{j+1}^v, \dots, f_m^v\}$ $(j = 1, \dots, m)$, that satisfies conditions: (1) $con(f_{i-1}^u, f_j^v)$, (2) $con(f_j^v, f_{i+1}^u)$, (3) $con(f_{j-1}^v, f_i^u)$, (4) $con(f_i^u, f_{j+1}^v)$, to get new permutation pairs:

$$\begin{aligned} \bar{p}^u &= \{f_1^u, \dots, f_{i-1}^u, f_j^v, f_{i+1}^u, \dots, f_n^u\} \\ \bar{p}^v &= \{f_1^v, \dots, f_{j-1}^v, f_i^u, f_{j+1}^v, \dots, f_m^v\} \end{aligned}$$

2. *Cut*: Given $p^u = \{f_1^u, \ldots, f_{i-1}^u, f_i^u, \ldots, f_j^u, f_{j+1}^u, \ldots, f_n^u\}$ $(i, j = 1, \ldots, n,\ i \neq j)$, then seek $p^v = \{f_1^v, \ldots, f_{k-1}^v, f_k^v, \ldots, f_l^v, f_{l+1}^v, \ldots, f_m^v\}$ $(k, l = 1, \ldots, m,\ k \neq l)$, that satisfies conditions: (1) $con(f_{i-1}^u, f_k^v)$, (2) $con(f_{k-1}^v, f_i^u)$, (3) $con(f_l^v, f_{j+1}^u)$, (4) $con(f_j^u, f_{l+1}^v)$, to get new permutation pairs:

$$\tilde{p}^u = \{f_1^u, \ldots, f_{i-1}^u, f_k^v, \ldots, f_l^v, f_{j+1}^u, \ldots, f_n^u\}$$
$$\tilde{p}^v = \{f_1^v, \ldots, f_{k-1}^v, f_i^u, \ldots, f_j^u, f_{l+1}^v, \ldots, f_m^v\}$$

3. *Insert*: Given $p^u = \{f_1^u, \ldots, f_{i-1}^u, f_i^u, f_{i+1}^u, \ldots, f_n^u\}$ $(i = 1, \ldots, n)$, then seek $p^v = \{f_1^v, \ldots, f_{k-1}^v, f_k^v, ..., f_l^v, f_{l+1}^v, \ldots, f_m^v\}$ $(k, l = 1, \ldots, m,\ k \neq l)$, that satisfies conditions: (1) $con(f_{i-1}^u, f_k^v)$, (2) $con(f_l^v, f_i^u)$, (3) $Des(f_{k-1}^v) = Ori(f_{l+1}^v)$, to get new permutation pairs:

$$\tilde{p}^u = \{f_1^u, \ldots, f_{i-1}^u, f_k^v, \ldots, f_l^v, f_i^u, \ldots, f_n^u\}$$
$$\tilde{p}^v = \{f_1^v, \ldots, f_{k-1}^v, f_{l+1}^v, \ldots, f_m^v\}$$

4. *Delete*: Given $p^u = \{f_1^u, \ldots, f_{i-1}^u, f_i^u, \ldots, f_j^u, f_{j+1}^u, \ldots, f_n^u\}$ $(i, j = 1, \ldots, n,\ i \neq j)$, if $con(f_{i-1}^u, f_{j+1}^u)$, then we can drop the inner substring and get the new string:

$$\tilde{p}^u = \{f_1^u, \ldots, f_{i-1}^u, f_{j+1}^u, \ldots, f_n^u\}$$

Note that for each of the neighborhood operator imposed on the flight string pair (p^u, p^v), there may be multiple new permutation pairs that satisfy the corresponding connection requirements. We greedily select the pair that leads to the minimum objective value based on the *Evaluator*. Through our experiments, this greedy selection strategy performs much better than random selection strategy regarding ultimate solution quality, with little sacrifice regarding overall runtime. In addition note that for the above operations to obtain new neighbors, not only is the connectivity of adjacent flights in p^u and p^v ensured, but also the origin and destination airport of each aircraft should remain the same as in the original schedule. Similarly, a pair of flight strings $q^u = \{f_1^u, \ldots, f_n^u\}$, $q^v = \{f_1^v, \ldots, f_m^v\}$ $(u, v \in U^{cr})$ are also randomly selected from Q, and undergo the same neighborhood operation as (p^u, p^v).

4.2.3. Distribution generator

In order to improve the sampling efficiency and effectiveness in Algorithm 2, we incorporate probability distributions over flight strings both in aircraft assignment and crew team assignment, based on features specific to the disruption scenario or state configuration. Accordingly, we generate the following three variants of VNS-based solution heuristics:

1. *U-VNS*: The probability distributions over flight strings of aircraft/crew team assignment are assumed to be uniform distributions.
2. *DT-VNS*: The distributions are related to the spatio-temporal *distance* to the disrupted flights. We obtain the shortest distance from each flight node to the nearest disrupted flight node in the flight network and take the negative value of the shortest distance as the distance attribute of the flight node. For each flight string, we compute the mean distance attribute over all the flights in the string and then perform min–max scaling followed by a softmax transformation to obtain the probability to sample this flight string. Accordingly, a flight string with elements experiencing disruptions (or being close to them) indicates a larger probability to be sampled, which is in accordance with the underlying idea of preprocessing techniques in Arıkan et al. (2017).
3. *DG-VNS*: The distributions are related to the degree of elements in the flight string concerning the flight network. We obtain the node degree of each flight node in the flight network, compute the mean degree for each flight string, and finally perform min–max scaling followed by a softmax transformation to obtain the probability to sample each flight string. In this manner, the flight string with higher-degree elements indicates a larger probability to be sampled, which is in line with the intuition that the flight string with higher connectivity may be more likely subject to neighborhood operations, e.g., swap and cut.

4.3. Deep reinforcement learning-based guidance

Given the large solution space of the airline disruption recovery problem in the real-world application, our proposed VNS-based heuristic does not scale well despite the fine-tuned neighborhoods, based on the following rationale. The VNS-based heuristic explores the neighborhood by a *fixed sequence*, even if some neighborhood operator has little contribution to the reduction of the objective (e.g., the delete operator is rarely used in experimental results due to the fact that cancelling a flight is costly). On the other hand, despite the guidance via a probability distribution over the flight strings, the VNS-based heuristic performs the sampling of flight string pairs in an *uninformed manner*, i.e., there is no feedback on how constructive/destructive the previous sampling action is, which can be characterized by *'zero learning'*.

The above discussion on the drawback of the VNS-based heuristic motivates us to incorporate Deep Reinforcement Learning (DRL) to prompt a more informed exploration process through the search space, and ultimately lead to high efficiency and effectiveness. In this framework, neighborhood operations and flight string pairs are selected based on the policies obtained by the interaction between the RL agent and the environment. Besides, these policies are learned ad-hoc for the to-be-solved problem at hand, thus the agent is well initialized in face of the emerging problem, without the need for further time-consuming exploration over the large search space, which meets the demand of airlines in seeking instant recovery solutions.

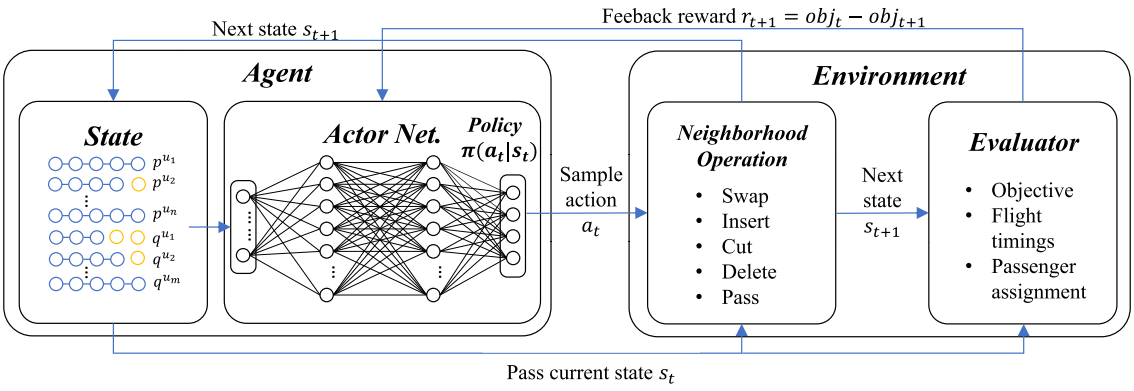

Fig. 2. The Deep Reinforcement Learning-based VNS guidance framework. The state is characterized by the aircraft assignment and crew team assignment, with blue circles referring to flight element indices, orange ones referring to padding zero. The current state is fed forward to an actor network and yields a stochastic policy, which is exploited to sample an action. The next state is generated in a deterministic way by the neighborhood operation based on the current state and action. The evaluator receives the next state and feeds back the reward (i.e., the change of objective value) to the actor network and the critic network (not displayed in the figure) to update their parameters. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Below, we first formulate our DRL-based VNS guidance framework as a Markov Decision Process (MDP). Then, we apply the state-of-the-art policy gradient method, Proximal Policy Optimization (PPO), to train our policy network for selecting neighborhood operations and flight string pairs.

4.3.1. MDP formulation

In this framework, the RL agent is employed to learn a policy that at each step, selects a pair of flight strings and a neighborhood operator for both aircraft and crew team to generate a new state. This process of learning is always performed in an *on-policy* manner, which means that each update of parameters only uses data collected while acting according to the most recent version of the policy (Achiam, 2018). Accordingly, this sequential decision problem is formulated as a discrete-time MDP. Specifically, we regard the policy (actor) network and the value (critic) network as the agents, while the neighborhood operator and evaluator as the environment. Our framework architecture is visualized in Fig. 2. In each episode of solving the airline disruption recovery problem, the MDP can be described as follows:

1. *States.* A state $s_t \in S$ is defined in a similar way as the one in Section 4.2, which is the combination of aircraft assignment and crew team assignment (P, Q). Due to the variable length of flight strings in P and Q, we build the state s_t as a two-dimensional tensor by substituting each flight element as an index and padding each flight string with zero based on the length of the longest flight string. Initially, the state s_0 is the tensor corresponding to the scheduled assignment of aircraft and crew team.
2. *Actions.* An action $a_t \in \mathcal{A}$ is defined to be the selection of a pair of flight string and a neighborhood operator for both aircraft and crew team at state s_t. Mathematically, the action can be expressed as a tuple $(p^{u_i}, p^{u_j}, k, q^{u'_m}, q^{u'_n}, k')$, where the neighborhood operator k is applied on flight string pair (p^{u_i}, p^{u_j}) selected from P while k' is applied on flight string pair $(q^{u'_m}, q^{u'_n})$ selected from Q. Note that $u_i \neq u_j$ and $u'_m \neq u'_n$. The size of action space can be directly obtained by combinatorics, i.e.,
$$|\mathcal{A}| = \binom{|U^{ac}|}{2} \cdot \binom{|U^{cr}|}{2} \cdot \mathcal{N}^2 \tag{23}$$
where $\mathcal{N}$ is the number of neighborhood operators. In this framework, we adopt five neighborhood operators: swap, insert, cut, delete and pass (i.e., remain the current pair of flight strings unchanged).
3. *Transition.* According to the neighborhood operation, the next state s_{t+1} is deterministically attained based on the current state s_t and action a_t. This state transition involves the override of a pair of flight strings for both aircraft assignment and crew team assignment.
4. *Rewards.* The reward function is defined as the change of objective value after state transition, i.e., $r_t = obj_t - obj_{t+1}$. This function turns into a penalty if the next-state objective is higher than the current-state objective. Given the step limit T_s for each episode, we adopt the finite-horizon discounted return from step t of the episode, $R_t = \sum_{k=t}^{T_s} \gamma^{k-t} r_k$, where $\gamma \in [0, 1]$ is the discount factor.
5. *Policy.* The stochastic parameterized policy $\pi_\theta(a_t \mid s_t)$ represents the conditional probability distribution over all the actions in the action space given a state s_t and the vector of all trainable weights θ. Since the learning process proceeds in an on-policy way, the agent starts from s_0, iteratively picks up an action a_t according to $\pi_\theta(\cdot \mid \cdot)$, transits to s_{t+1} and obtains the reward r_t, until reaching the step limit T_s. This forms a trajectory $\tau = \{s_0, a_0, r_1, \ldots, s_{T_s-1}, a_{T_s-1}, r_{T_s}\}$ for each episode.

4.3.2. Training method

We apply a state-of-the-art policy gradient method, Proximal Policy Optimization (PPO) to train our policy network. Methods that learn approximations to both policy function and value function are called actor-critic methods, where *actor* refers to the learned policy, and *critic* is a reference to the learned value function, e.g., a state value function. PPO is motivated by the trade-off between taking the biggest possible improvement step on a policy based on current data and avoiding to accidentally cause performance collapse, thus exploiting a family of first-order methods to keep new policies close to old (Schulman et al., 2017). In our framework, we utilize the PPO-clip variant which relies on specialized clipping in the loss function to remove incentives for the new policy to get far from the old policy (Achiam, 2018). Due to space limitation, we leave the training algorithm to Appendix B.

5. Experimental evaluation

In the following, we perform a rich set of computational experiments to highlight the efficiency and effectiveness of our approach. First, Section 5.1 provides a description of the generation of the airline operational datasets. Section 5.2 reports the related (hyper) parameters in the VNS-based heuristic and DRL-based VNS guidance framework. Section 5.3 compares the efficiency and efficacy of our proposed heuristic with respect to the exact CPLEX solution, and Section 5.4 further demonstrates the scalability of the heuristic for large-scale instances. Section 5.5 discusses the potential of our DRL-based framework in achieving Transfer Learning (TL), i.e., applying the pre-trained model to obtain recovery solution efficiently.

5.1. Underlying airline operational dataset

The airline operational datasets used in this study are generated according to a set of well-designed algorithms, which satisfy most of the restrictions for entities in airline operations. Each airline operational dataset contains a flight schedule, a passenger itinerary table, and a disruption scenario (see Section 3.1.1). We use the airports dataset from OurAirports (2022), which provides the name, identity, latitude, longitude and country of over 50,000 airports. Besides, we use the aircraft characteristics dataset from Federal Aviation Administration (2018), which provides the capacity data of 50 aircraft models. We select a set of airports in the contiguous United States with top flow volume and generate a connectable graph whose nodes are the selected airports and edges are possible flight links which satisfy distance constraints in practice.

We provide the algorithms that are applied to generate flight schedules, crew routing and passenger itineraries in Appendix C. The flight schedule table contains the following information for each flight of the airline: tail number, flight, crew team, departure airport, destination airport, schedule time of departure, schedule time of arrival, flight time, cruise time, flight distance, tail capacity and number of passengers on the flight. The passenger itinerary table contains the following information for each itinerary of the airline: flight(s) taken by each itinerary, departure airport, destination airport, scheduled time of departure for the itinerary, scheduled time of arrival for the itinerary, number of passengers of the itinerary.

Here, we report the values of parameters used in this study to generate airline operational datasets. The number of aircraft (trajectories) and the number of airports directly lead to the relative size of the dataset, so we report them in the following sections. The number of unique aircraft types is set to 3, each with a load factor of 0.8. The distance of a valid flight is limited to the range [600, 3000] km. The average speed of aircraft used to estimate flight duration is set to 800 $\mathrm{km/h}$. The minimum connection time for aircraft, crew team and passengers is set to 30 min. The airline operations start at 5 AM and end at 2 AM the next day. The cruise time for a flight is assumed to be 30 min shorter than the entire flight time. The maximum departure/arrival holding time is set to 2 h, which is related to the latest departure or arrival time of the flight. The minimum and maximum connection times for aircraft are set to 30 min and 600 min respectively. The minimum and maximum sit times for crew members are set to 30 min and 240 min respectively. The maximum number of landings within a crew duty is set to 4. The maximum flying time within a duty is set to 8 h, and the maximum duty duration is set to 12 h (Haouari et al., 2019). The minimum and maximum transfer time for passengers are set to 30 min and 480 min respectively. The fraction of direct/two-hop/three-hop itineraries are set to 0.85, 0.12 and 0.03 respectively. The maximum number of connecting flights taken by passengers is restricted to 3.

As for the parameters related to cost, the flight cancellation cost is assumed to be $\$20,000$ per flight (Marla et al., 2017); jet fuel price is assumed to be $\$0.478/lb$ (Marla et al., 2017); delay cost is assumed to be $\$1.0242$ per passenger per minute (Ball et al., 2010b). The negative cost to prompt aircraft and crew teams to follow their original schedule is set to $\$1$ per flight, and this negative cost for the passengers is set to $\$0.01$ per flight.

As for disruption scenarios, we consider two types of scenarios based on the severity of disruption: a mild disruption scenario (denoted as DS^-) represents a scenario where 10% of the flights are disrupted/delayed, while a severe disruption scenario (denoted as DS^+) represents a scenario where 30% of the flights are disrupted/delayed. This delay of schedule departure time is assumed to be a random variable that follows a uniform distribution over $[0, 240]$ min. We visualize the dataset $AC8$ which contains eight aircraft and eight airports both on geographical map (Fig. 3) and time–space network (Fig. 4).

5.2. Parameter selection

In the following, we report the (hyper) parameters of the above solution methods, including the CPLEX-based exact solution, VNS-based heuristic solution, and DRL-based VNS guidance heuristic solution. Our experiments are performed on a remote server with 6 Intel Xeon Gold 6142 CPU, RTX 3080 GPU with 10.5 GB GPU memory and 27 GB RAM, installed with IBM ILOG CPLEX

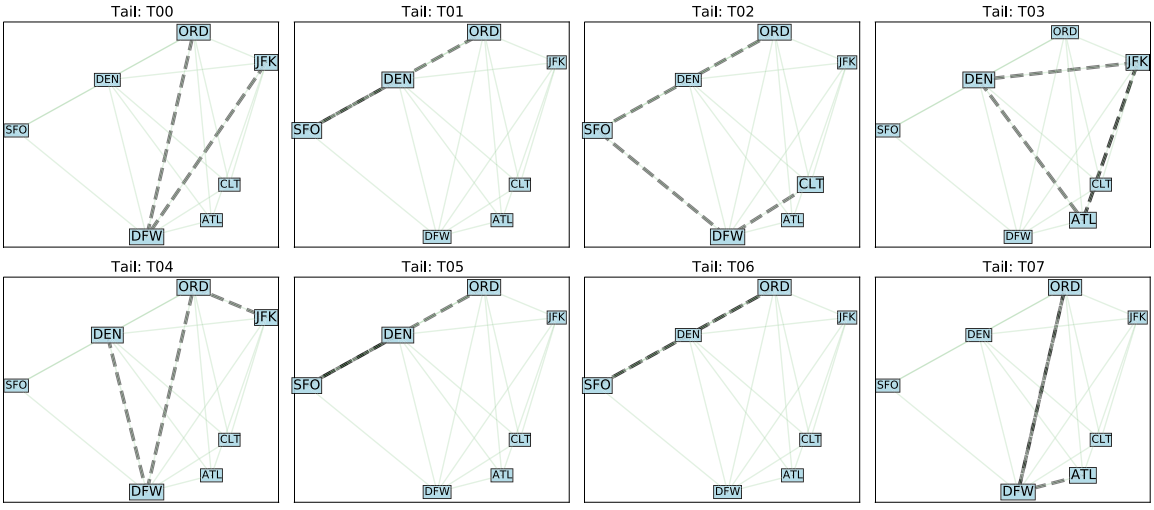

Fig. 3. The aircraft trajectories of dataset $AC08$ visualized on a geographical map. The dataset contains eight aircraft (T00 to T07) and eight airports (the nodes labeled by IATA codes). In each subplot, the consecutive dashed links represent the trajectory of the aircraft. Airports that are passed in the trajectory are highlighted.

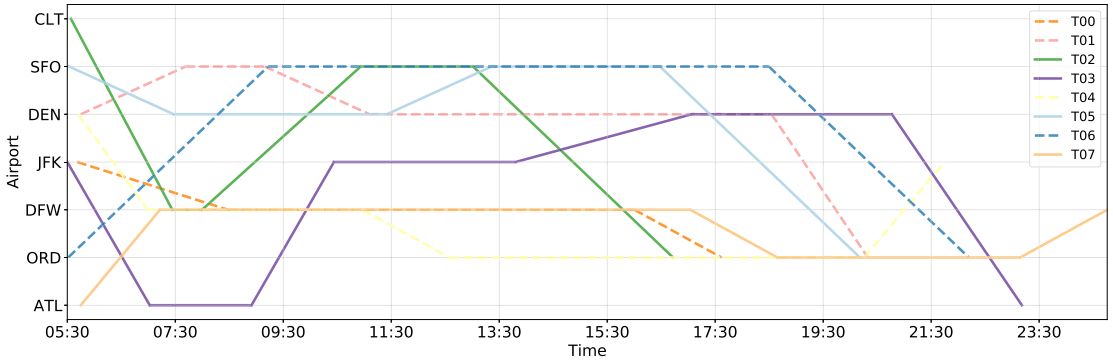

Fig. 4. The aircraft trajectories of dataset $AC08$ visualized in time–space network. Each polyline represents the trajectory of an aircraft with respect to time (abscissa axis) and space (ordinate axis).

Optimization Studio v12.10 for Linux and PyTorch v1.10. Our code is available on github.[1] The related (hyper) parameters of each method are reported below.

- CPLEX: We take into account the variation of cruise speed and thus the additional fuel cost in CPLEX computation. The time limit of the CPLEX solver is set to 1000 s to generate the optimal or close-to-optimal solution for each instance.
- VNS: We set the length of trajectory for each VNS baseline $\mathcal{L}_T = 10$ since an optimal solution is generally not far from the initial solution in the search space through our experiments and a deeper search may not be helpful to improve the solution quality to some extent. The number of neighborhoods $\mathcal{N} = 5$, including swap, insert, cut, delete and pass.
- DRL: We adopt the same structure for the actor network and the critic network. The input layer of the artificial neural network receives the current state as a flattened array, feeds forward to two hidden layers each with 256 neurons and ReLU as activation function, and the output layer with Softmax as activation function yields the probability distribution of actions. The agent is trained using the Adam optimizer and a static learning rate $\alpha = 10^{-5}$. The length of trajectory is set to $\mathcal{L}_T = 10$ as VNS baselines. The reward discount is set to $\gamma = 0.9$ and the smoothing parameter is set to $\lambda = 0.95$ as Schulman et al. (2017). We use $\epsilon = 0.8$ for probability clipping which encourages exploration but also avoids stepping too far from current policy. The number of training epochs is set to $K = 3$. The episodes to train each instance range from 2000 (for small instances such as $AC05$) to 10000 (for large instances such as $AC300$).

[1] https://github.com/Yida-Ding/ADM-Research

Table 8
Statistics of small/medium-scale datasets and disruption scenarios. For each dataset, the number of flights, airports, aircraft, crew teams, itineraries and passengers are listed. For each disruption scenario, the number of delayed flights, flights with delay less than two hours, flights with delay longer than two hours are listed.

Dataset	Dataset Features						DS^- Features			DS^+ Features		
	$\|F\|$	$\|A\|$	$\|U^{ac}\|$	$\|U^{cr}\|$	$\|U^{it}\|$	$\|U^{ps}\|$	$\leq 2h$	$> 2h$	$\|D^f\|$	$\leq 2h$	$> 2h$	$\|D^f\|$
$AC05$	16	4	5	6	17	2020	0	1	1	2	2	4
$AC10$	28	8	10	12	30	3731	1	1	2	7	1	8
$AC15$	42	12	15	21	47	5359	3	1	4	4	8	12
$AC20$	62	20	20	26	67	7416	4	2	6	10	8	18
$AC25$	73	21	25	34	88	8638	4	3	7	10	11	21
$AC30$	94	28	30	37	112	11 415	3	6	9	16	12	28

Table 9
Comparison of the objective and runtime among PPO-VNS, CPLEX and VNS baselines in small/medium-scale instances. The CPLEX relative optimality gaps (cgap) output by the solver are reported.

Instance	PPO-VNS		U-VNS		DT-VNS		DG-VNS		CPLEX		
	obj.	time	obj.	time	obj.	time	obj.	time	obj.	time	cgap
$AC05DS^-$	23 534.7 (**0.57%**)	0.022	24 449.6 (4.48%)	0.068	24 205.0 (3.43%)	0.092	24 352.4 (4.06%)	0.108	23 401.2	79	0.00%
$AC05DS^+$	46 975.5 (**0.00%**)	0.016	51 254.6 (9.11%)	0.073	52 030.5 (10.76%)	0.098	51 270.2 (9.14%)	0.099	46 975.5	1001	0.01%
$AC10DS^-$	30 973.3 (**0.66%**)	0.019	58 128.5 (88.91%)	0.095	45 244.4 (47.03%)	0.092	57 234.6 (86.00%)	0.098	30 771.2	22	0.00%
$AC10DS^+$	55 558.7 (**0.12%**)	0.022	55 925.1 (0.78%)	0.102	55 899.2 (0.73%)	0.125	55 920.8 (0.77%)	0.113	55 493.6	1003	0.02%
$AC15DS^-$	44 557.9 (**0.63%**)	0.047	46 124.2 (4.16%)	0.089	46 124.2 (4.16%)	0.094	46 219.7 (4.38%)	0.097	44 281.1	1002	0.01%
$AC15DS^+$	194 732.6 (**0.05%**)	0.049	195 256.2 (0.32%)	0.096	195 210.4 (0.30%)	0.087	195 120.3 (0.25%)	0.099	194 630.3	1004	0.03%
$AC20DS^-$	94 751.1 (**0.00%**)	0.091	96 580.7 (1.93%)	0.089	96 702.9 (2.06%)	0.099	96 479.0 (1.82%)	0.107	94 751.1	53	0.00%
$AC20DS^+$	263 584.7 (**1.08%**)	0.094	265 366.9 (1.76%)	0.112	265 464.5 (1.80%)	0.128	265 562.1 (1.83%)	0.178	260 781.2	111	0.00%
$AC25DS^-$	89 638.7 (**1.46%**)	0.125	91 458.2 (3.52%)	0.242	91 458.2 (3.52%)	0.223	91 325.8 (3.37%)	0.289	88 348.0	24	0.00%
$AC25DS^+$	308 576.0 (**0.37%**)	0.094	312 354.0 (1.60%)	0.257	321 023.5 (4.42%)	0.212	321 745.4 (4.66%)	0.267	307 424.4	1009	0.07%
$AC30DS^-$	191 884.8 (**0.64%**)	0.147	191 884.8 (0.64%)	0.335	191 884.8 (0.64%)	0.328	190 586.8 (0.64%)	0.359	190 662.1	1010	0.02%
$AC30DS^+$	357 254.8 (**0.03%**)	0.052	368 408.5 (3.15%)	0.231	368 687.5 (3.23%)	0.289	368 579.2 (3.20%)	0.256	357 154.7	1011	0.04%

5.3. Comparison with exact method

In the following, we compare our proposed heuristic solutions with the exact CPLEX solution regarding objective value and runtime to demonstrate the efficiency and effectiveness of our proposed methodology. We generate small/medium-scale datasets $AC05$ (five aircraft trajectories) to $AC30$ (thirty aircraft trajectories) and impose two disruption scenarios DS^- and DS^+ to each dataset. The related statistics of the datasets and scenarios are reported in Table 8, including the number of flights, airports, aircraft, crew teams, itineraries and passengers for each dataset, and the number of delayed flights, flights with delay less than two hours, flights with delay longer than two hours for each scenario. The number of entities grows with respect to the number of aircraft and flights, and a rapid-growing number of decision variables will emerge in CPLEX computation, which leads to inefficiency for airlines to make instant recovery decisions.

The experimental results regarding the comparison of PPO-VNS, CPLEX and VNS baselines are summarized in Table 9. For each of the 12 instances, we set the runtime limit of the CPLEX solver to be 1000 s and we report the overall runtime including flight network preprocessing time and CPLEX runtime. The majority of instances exhaust the runtime limit and yield relative optimality gaps (cgap in Table 9), while five instances obtain the optimal solution and terminate early. All of the VNS-based methods are run for 100 times and we report the average objective value and runtime. As for the PPO-VNS method, the agent is pre-trained and we report the average testing objective value and runtime in Table 9. The number of training episodes ranges from 400 for smaller instance $AC05DS^-$ to 1300 for larger instance $AC30DS^+$. The training time ranges from ten seconds for smaller instance $AC05DS^-$ to 70 s for larger instance $AC30DS^+$. We compute the gap of objective value for each method with respect to the (close-to) optimal objective value obtained by CPLEX. Compared to the CPLEX results, the PPO-VNS method greatly reduces the runtime, which meets the need of airlines in seeking instance solution; on the other hand, the gap between PPO-VNS and CPLEX stay within 1.5% and is smaller than the gap between VNS baselines and CPLEX, which demonstrates the accuracy and reliability of the deep reinforcement learning-based approach. Due to the potential limitation of the evaluator embedded in all VNS-based methods, the objective values of VNS-based methods cannot well approach the (close-to) optimal objective value generated by CPLEX, but the gap is small for the majority of instances and exactly zero for a minority of instances (e.g., $AC05DS^+$ and $AC20DS^-$). Regarding the comparison among three VNS-baselines, DT-VNS and DG-VNS generally achieve smaller objective values than U-VNS, which reveals the effect of the guidance based on probability distribution.

In the training process of our experiments, PPO demonstrates great capability in efficiently exploring the search space and avoiding getting stuck in a local minimum. We visualize the convergence process of objective values of the PPO-VNS method for each instance in Fig. 5, as compared with the levels of objective values generated by CPLEX and VNS baseline methods. For each instance, the abscissa is the training episode while the ordinate is the moving average of objective values in 100 episodes. The PPO agent is uninitialized at the first episode (i.e., the initial weights and biases of actor and critic network are randomly determined),

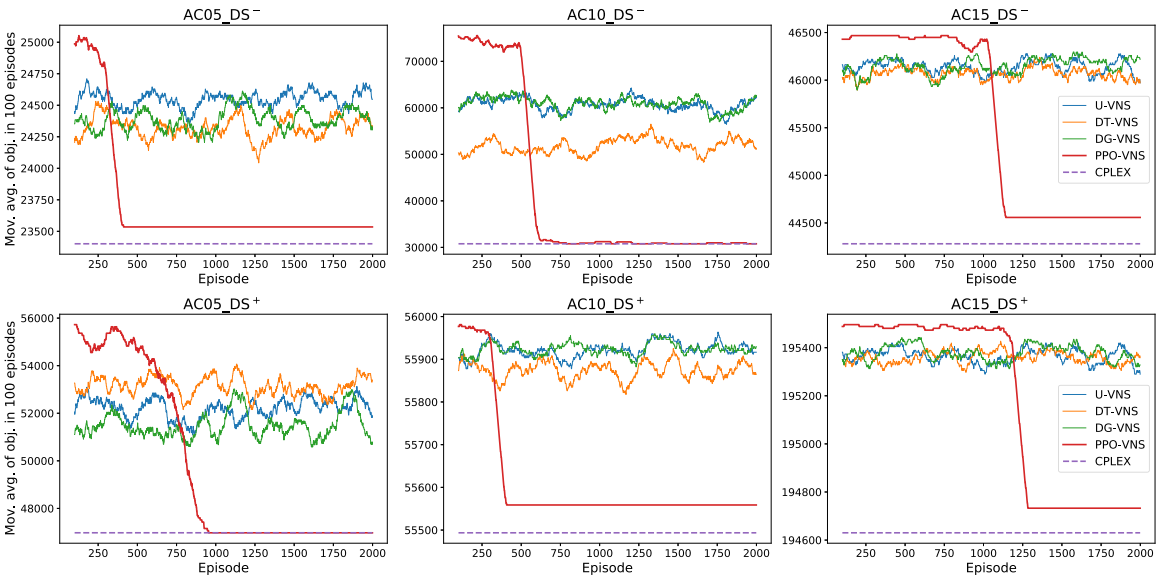

Fig. 5. The convergence process of objective values of PPO-VNS method in each instance, with comparison to the levels of objective values generated by CPLEX and VNS baseline methods. For each instance, the abscissa is the training episode while the ordinate is the moving average of objective values in 100 episodes.

Table 10
Comparison of the objective and runtime among PPO-VNS and VNS baselines in large-scale instances. The results on runtime demonstrate the scalability of our proposed heuristic.

Instance	Feature		PPO-VNS		U-VNS		DT-VNS		DG-VNS	
	$\|F\|$	$\|D^f\|$	obj.	time	obj.	time	obj.	time	obj.	time
$AC100DS^-$	306	30	194 927.5	1.382	213 960.1 (9.76%)	1.587	213 253.5 (9.40%)	2.134	213 388.8 (9.47%)	1.748
$AC100DS^+$	306	91	572 589.1	1.872	621 591.8 (8.56%)	2.036	619 305.9 (8.16%)	2.755	618 519.1 (8.02%)	1.759
$AC150DS^-$	453	45	391 816.4	2.077	422 911.9 (7.94%)	2.685	423 236.4 (8.02%)	3.294	423 120.4 (7.99%)	2.653
$AC150DS^+$	453	135	1 192 954.6	2.968	1 219 018.2 (2.18%)	2.853	1 218 473.9 (2.14%)	4.528	1 218 908.1 (2.18%)	2.717
$AC200DS^-$	603	60	630 277.8	4.120	662 758.5 (5.15%)	4.567	663 470.2 (5.27%)	6.251	662 820.9 (5.16%)	4.935
$AC200DS^+$	603	180	1 465 999.2	5.331	1 493 812.6 (1.90%)	4.680	1 493 725.6 (1.89%)	7.910	1 494 024.9 (1.91%)	5.067
$AC250DS^-$	770	77	634 143.8	6.897	641 716.9 (1.19%)	8.500	641 840.5 (1.21%)	10.618	641 832.2 (1.21%)	7.899
$AC250DS^+$	770	231	1 958 051.1	8.755	1 974 541.9 (0.84%)	8.456	1 974 676.5 (0.85%)	13.403	1 974 086.2 (0.82%)	8.632
$AC300DS^-$	931	93	814 248.2	9.660	833 174.1 (2.32%)	10.301	833 203.1 (2.33%)	12.804	833 282.3 (2.34%)	10.879
$AC300DS^+$	931	279	2 422 490.8	14.091	2 479 091.2 (2.34%)	9.508	2 479 126.1 (2.34%)	16.924	2 479 500.6 (2.35%)	10.249

leading to a relatively weaker performance compared to VNS baselines. As the training episodes pass by, the PPO agent manages to learn the critical trajectories to significantly reduce the objective value, while the VNS baselines remain the same levels without learning. Eventually, the objective values of the PPO-VNS method keep close to the optimal objective obtained by CPLEX or even coincide with it.

5.4. Scalability of heuristics

Based on the previous results on small/medium-scale instances, we further conduct experiments for the VNS-based methods on the large-scale instances to verify the scalability of our proposed methodology. In our experiments, the large-scale instances (e.g., $AC300$) lead to a combinatorial challenge for the CPLEX solver to obtain an optimal solution due to the intractable number of decision variables especially flow variables indicating the assignment of flights to entities. As for the deep reinforcement learning-based method, the size of the action space also grows significantly with respect to the increase of $|U^{ac}|$ and $|U^{cr}|$. To avoid an overflow of GPU memory, we further purpose to limit the size of action space based on the aforementioned distance metric. Specifically, for the experiments on $AC100$ to $AC300$, we precompute the probability distribution of distance metric over all the flight strings for aircraft/crew teams based on the scheduled assignment, and select the top 20 most probable flight strings for aircraft assignment and top 10 most probable flight strings for crew team assignment. This size control of action space ensures the feasibility of the PPO-VNS method without sacrificing too much on the optimality of objective value.

We report our experimental results regarding the scalability of heuristics in Table 10. All of the VNS-based methods are run for 100 times and we report the average objective value and runtime. As for the PPO-VNS method, the agent is pre-trained and we

Table 11
Comparison of the performance regarding transfer learning, uninitialized learning and VNS baselines. For each instance, the objective value after convergence and the critical episode that the objective function converges for both transfer learning method and uninitialized learning (i.e., train/run the PPO-VNS agent from scratch) are listed, as compared with the average objectives of VNS baselines in 100 iterations.

Instance	Feature		Transfer learning		Uninitialized learning		VNS baseline objective		
	$\|F\|$	$\|D^f\|$	obj.	eps.	obj.	eps.	U-VNS	DT-VNS	DG-VNS
$AC05DS_0$	16	1	27 503.2 (00.00%)	272	27 503.2	1541	27 964.8 (08.95%)	27 720.0 (08.00%)	27 769.5 (08.19%)
$AC05DS_1$	16	1	27 434.8 (00.00%)	127	27 434.8	1070	31 860.7 (16.13%)	31 252.7 (13.92%)	31 473.8 (14.72%)
$AC05DS_2$	16	1	9051.6 (00.00%)	0	9051.6	0	9051.6 (00.00%)	9051.6 (00.00%)	9051.6 (00.00%)
$AC10DS_0$	28	2	26 131.3 (00.01%)	1041	26 133.3	1554	31 781.2 (21.62%)	31 036.3 (18.77%)	31 781.3 (21.62%)
$AC10DS_1$	28	2	28 407.7 (00.00%)	1150	28 407.7	2294	31 905.6 (12.31%)	31 406.0 (10.55%)	31 905.6 (12.31%)
$AC10DS_2$	28	2	59 568.4 (00.00%)	0	59 568.4	3349	59 568.4 (17.91%)	59 387.5 (17.55%)	59 477.9 (17.73%)
$AC15DS_0$	42	4	91 314.6 (−02.58%)	0	93 737.0	1939	101 645.1 (08.44%)	101 357.3 (08.13%)	101 913.0 (08.72%)
$AC15DS_1$	42	4	37 299.2 (00.00%)	0	37 299.2	0	37 299.2 (00.00%)	37 299.2 (00.00%)	37 299.2 (00.00%)
$AC15DS_2$	42	4	15 194.9 (00.00%)	0	15 194.9	1295	18 299.8 (20.43%)	18 231.5 (19.98%)	18 436.3 (21.33%)
$AC20DS_0$	62	6	79 712.7 (00.00%)	609	79 712.7	3542	90 700.6 (13.78%)	90 700.6 (13.78%)	90 342.3 (13.33%)
$AC20DS_1$	62	6	81 179.0 (00.00%)	401	81 179.0	472	81 060.0 (07.76%)	81 060.0 (07.76%)	81 119.5 (07.83%)
$AC20DS_2$	62	6	120 448.4 (00.00%)	533	120 448.4	594	135 129.9 (12.19%)	135 277.8 (12.31%)	135 302.0 (12.33%)

report the average testing objective value and runtime. Due to the absence of the optimal solution for each instance, we compute the gap of objective for each method based on the smallest mean objective generated by all the methods. For all the instances in Table 10, the PPO-VNS method generates the minimum mean objective as compared to other baselines. The runtime statistics are generally comparable for all VNS-based methods, which demonstrates the scalability of our proposed approach.

5.5. Transfer learning

Based on the previous results, the trained PPO-VNS agent can solve the airline disruption recovery problem efficiently and reliably, however, the training process, especially for large-scale cases can take long time compared to the VNS baselines without learning. On the other hand, the flight schedule of the airline, passenger itinerary and other scheduled assignments are all available before the day of operation. These considerations motivate us to apply *transfer learning* to save the training time and further improve recovery solution quality. Transfer learning (TL) is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem (West et al., 2007). In our framework, instead of training the actor network and critic network from scratch, we can pre-train both networks by utilizing the dataset (flight schedule, itinerary, etc.) at hand and a set of artificial disruption scenarios. These artificial disruption scenarios can be either obtained by domain knowledge/experience or a random event generator, long before the actual disruption occurs. With certain expertise on the features of the dataset and possible disruption scenarios, the pre-trained PPO-VNS agent will be better initialized to solve the recovery problem.

In the following, we perform a set of experiments to verify the feasibility and advantage of transfer learning. For each small/medium-scale dataset $AC05$ to $AC20$, we pre-train the PPO-VNS agent with ten artificial disruption scenarios. Note that these artificial disruption scenarios are on top of the same airline operational dataset (flight schedule and passenger itinerary), thus they share the same flight network structure and can be interacted by the same PPO-VNS agent, i.e., the PPO-VNS agent is network-specific in this case. These ten artificial disruption scenarios are sampled by the same distribution, i.e., 10% of the flights experience a delay that is uniformly distributed over [0, 240] min. We train the PPO-VNS agent for 500 episodes in each artificial disruption scenario in a sequential manner, in an attempt to avoid overfitting on specific training scenarios and enhance generalization. The training time ranges from 110 s for smaller dataset $AC05$ to 260 s for larger dataset $AC30$. In the testing stage, we transfer the pre-trained agent to three testing scenarios (denoted as DS_0, DS_1 and DS_2) respectively, which are sampled by the same distribution as training scenarios, and keep optimizing the objective until convergence.

Comparison of the performance regarding transfer learning, uninitialized learning and VNS baselines are reported in Table 11. For each instance, the objective value after convergence and the critical episode that the objective function converges for both transfer learning method and uninitialized learning (i.e., train/run the PPO-VNS agent from scratch) are listed, as compared with the average objectives of VNS baselines in 100 iterations. We calculate the gap of objective values for each method with respect to the objective of the uninitialized learning method, as reported in the parenthesis. The majority of objective values for the transfer learning method coincide with that of regular uninitialized learning, and the transfer learning method yields a better objective in $AC15DS_0$ by 2.58%. As for the convergence of objective function, the transfer learning method always converges faster than uninitialized learning in all the instances, and in five of the instances, the agent is initialized with a perfect policy/value function that directly leads to best convergence. Besides, the transfer learning method demonstrates better solution quality as compared to the VNS baselines with a 10% gap on average. Besides, the effect of transfer learning reduces as the size of instance becomes larger. The reason behind this fact may be that the similarity between training instances and testing instances become lower as the size of the instance increases, since the underlying flight network of the instance becomes more complicated.

We visualize the convergence process of objective values of the transfer learning and the uninitialized learning for each instance in Fig. 6, as compared with the levels of objective values by VNS baseline methods. For each instance, the abscissa is the training episode while the ordinate is the moving average of objective values in 100 episodes. For the instances $AC10DS_0$, $AC10DS_1$ and

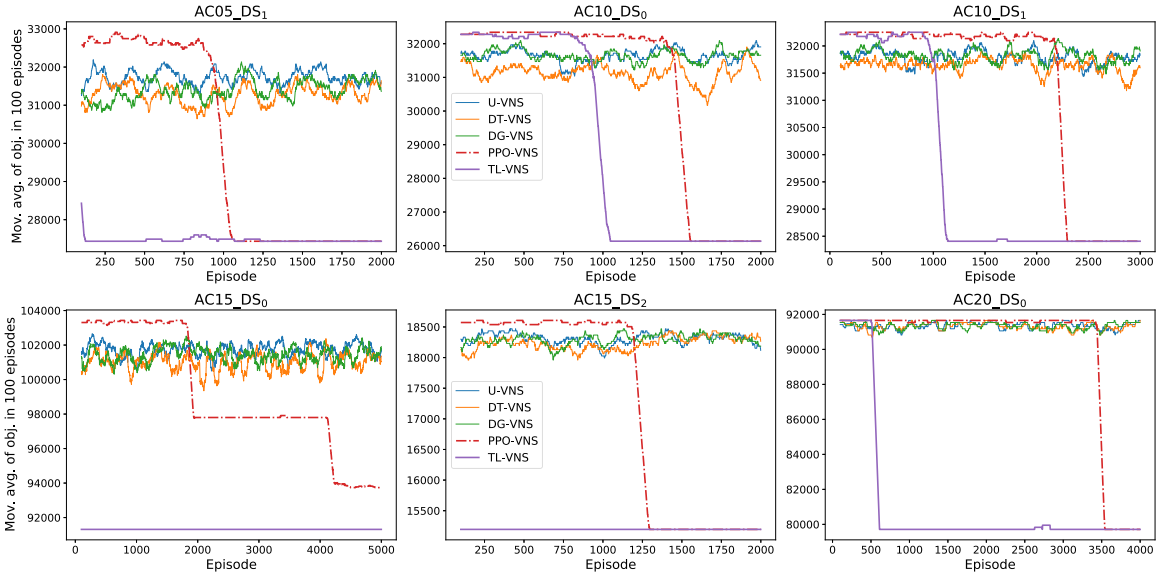

Fig. 6. The convergence process of objective values of the transfer learning and the uninitialized learning in each instance, with comparison to the levels of objective values by VNS baseline methods. For each instance, the abscissa is the training episode while the ordinate is the moving average of objective values in 100 episodes.

$AC20DS_0$, the objective values of the transfer learning and uninitialized learning are comparable at the beginning episode, but the objective function of transfer learning converges much sooner than uninitialized learning (for $AC20DS_0$ there is an advance of about 3000 episodes), which verifies that transfer learning can leverage knowledge (features, weights, etc.) from pre-trained policy/value functions and make the convergence process faster than training from scratch. As for instances $AC05DS_1$, $AC15DS_0$ and $AC15DS_2$, the transfer learning agent is initialized with perfect policy/value functions that directly lead to the best objective value.

6. Conclusions

In this paper, we formulate a decision recommendation framework for the integrated airline recovery problem and perform an in-depth investigation of the solution methodology regarding how to solve the problem efficiently and effectively for small/medium/large-scale instances. The underlying model is expressive in being able to incorporate aircraft and crew rerouting, passenger reaccommodation, departure holding, flight cancellation, while considering aircraft cruise speed decisions. Given the computational challenge of solving the model, we develop three solution techniques, the CPLEX-based exact solution technique with adequate preprocessing techniques, the VNS-based solution heuristic with well-designed neighborhood operators and state evaluator, and the novel DRL-based VNS guidance framework with pre-trained policy and value function based on transfer learning. In the proposed framework, we formulate the sequential decision problem as a discrete-time MDP in which the DRL agent feeds forward the current state (i.e., aircraft and crew team assignment) to a policy network and yields a stochastic policy, which is exploited to sample an action, then the next state is generated by the neighborhood operation and evaluated by the evaluator leading to reward signal. The training algorithm PPO is utilized to train the policy (actor) network and value (critic) network. Experimental results demonstrate the efficiency and effectiveness of our proposed PPO-VNS method. The PPO-VNS scales well in solving large-scale instances with adequate size control of action space. Experiments on transfer learning demonstrate that the pre-trained DRL agent can leverage features/weights from the training instances to accelerate the arrival of objective convergence and improve solution quality.

We acknowledge that our flight network-based model is subject to several limitations and we encourage future research to focus on them, summarized below: (1) *Aircraft maintenance*. We model aircraft maintenance as a must-node in the network based on Arıkan et al. (2017), Gürkan et al. (2016), indicating that it is a mandatory event that must occur in the schedule. This must-node approach can be useful for modeling preventive maintenance, where aircraft must be taken out of service for a scheduled maintenance check. In the aircraft recovery literature, researchers also use flexible time windows approach to model maintenance activities. Future studies may focus on extending the model by flexible time windows approach. (2) *External links*. We recommend future studies focus on the implementation of external flight links, which cover the recovery actions including aircraft ferrying, crew deadheading and reaccommodation of passengers to other transportation modes. A systematic approach to generate external arcs rather than by experience is crucial to the actual deployment of the model. (3) *Competing airlines*. Our approach could potentially have an impact on the schedules of other airlines if the recovery plan involves the use of shared resources such as runways, gates, and air traffic control systems. This could be further explored in future research.

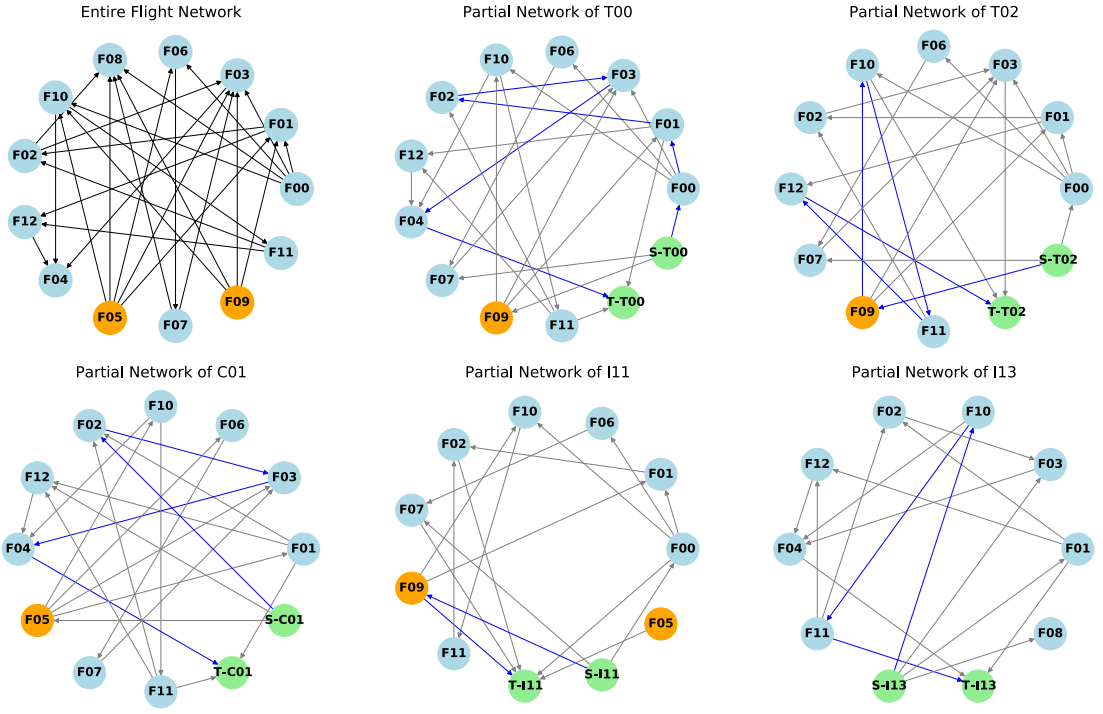

Fig. A.7. The flight network representation of the example (top left) and the partial flight networks for entity T00, T02, C01, I11 and I13. In the entire flight network, the possible connection links (E) are visualized by black arrows. The orange nodes are the flight nodes experiencing disruptions due to external reasons while the blue nodes are the flights that follow the schedule. In partial flight networks, the green nodes are source/sink nodes specific to entities. The blue arrows emanating from the source node, passing through flight nodes, and finally directing to sink nodes represent the scheduled routing of the entity, while the grey arrows represent the possible connection links (i.e., rerouting options). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

CRediT authorship contribution statement

Yida Ding: Conceptualization, Software, Methodology, Writing – original draft, Writing – review & editing. **Sebastian Wandelt:** Conceptualization, Methodology, Validation, Writing – review & editing, Supervision. **Guohua Wu:** Conceptualization, Methodology, Validation, Writing – review & editing. **Yifan Xu:** Methodology, Writing – review & editing. **Xiaoqian Sun:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition, Project administration.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: The authors report financial support was provided by National Natural Science Foundation of China.

Acknowledgments

This study is supported by the National Natural Science Foundation of China (Grant No. U2233214, No. 62250710166).

Appendix A. Synthesized example in Section 3.3

In the following, we provide more information and draw more insights from the recovery solution of the example shown in Section 3.3. In order to generate all possible recovery actions for entities (e.g., rerouting and swapping), we consider the flight network representation of this example here. The entire flight network is generated according to Eq. (1), while the *partial flight networks* specific to entities are the subsets of the entire flight network. More specifically, for each entity $u \in U$, the node set $\mathcal{V}^u$ of the partial network is the intersection of two node sets, i.e., the set of descendants of s^u and the set of ancestors of t^u; extracting the links among $\mathcal{V}^u$ will yield the link set $\mathcal{E}^u$ of the partial network. The flight network representation of this example and the partial flight networks for several entities are visualized in Fig. A.7. The grey arrows in these partial networks represent possible rerouting actions, including aircraft/crew swapping/rerouting and passenger reaccommodation.

To gain more insights from the recovery plan, the flight string representations of the original schedule and optimal recovery plan visualized in Fig. A.8 are created. Since F09 experiences severe departure delay, rerouting passengers from F09 to the best candidate

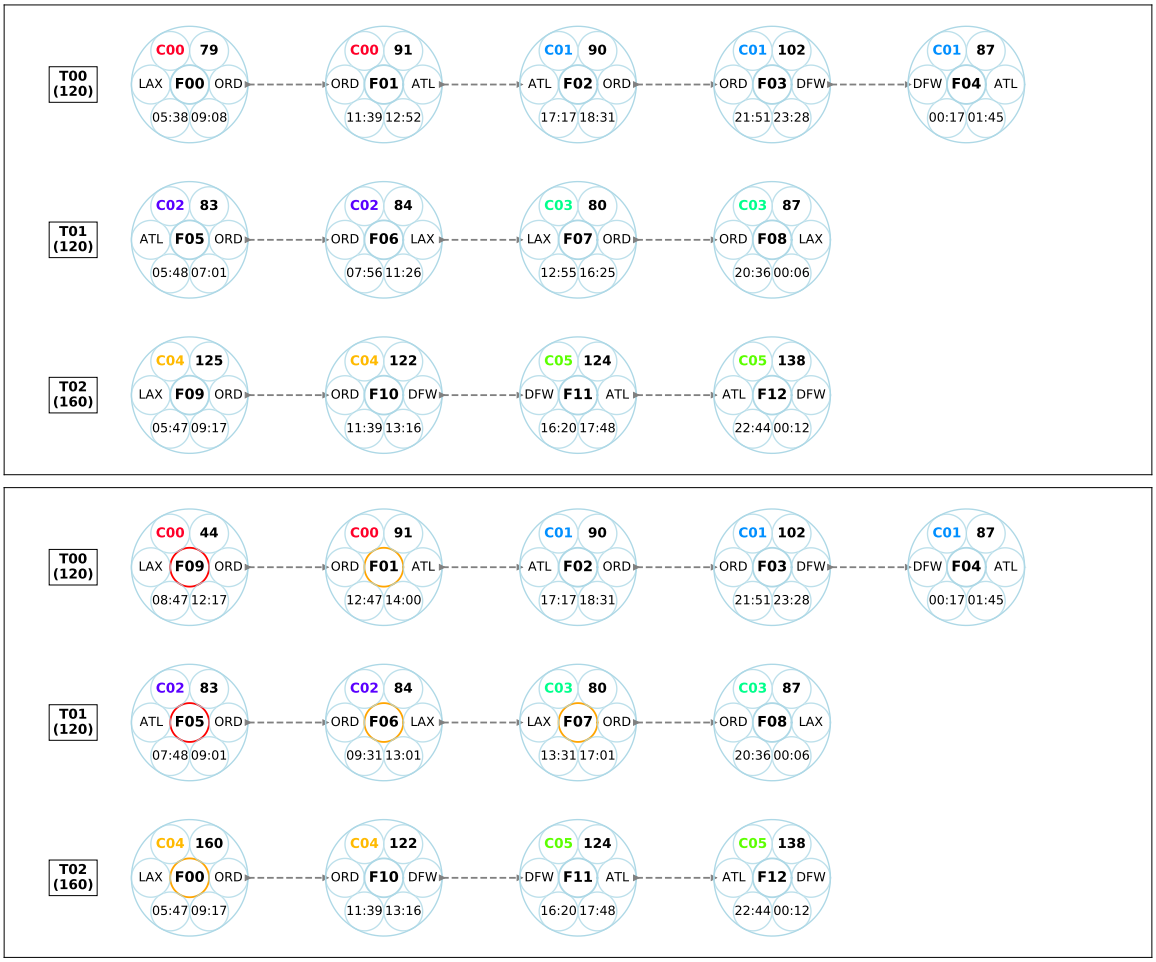

Fig. A.8. The flight string representations of the original schedule (top) and optimal recovery plan (bottom) for the synthesized example. Each large blue circle represents a flight node, with the related information (crew team, number of passengers, origin–destination pair and departure/arrival times) surrounding the middle circle of the flight index. Each flight string represents the flight assignment to specific aircraft, which is shown in the black box together with its seat capacity. A flight node with a red middle circle refers to the experience of disruption due to external reasons, while a flight node with an orange middle circle refers to the experience of propagated flight delay. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 3 PPO-based algorithm

Input: # of timesteps in each episode T_s, # of epochs K
Output: Trained actor network (θ) and critic network (ϕ)

```
for each episode do
    Run policy π_{θ_old} in environment for T_s timesteps
    Compute advantage estimates Â_t
    Optimize loss function L^{PF+VF} wrt. θ and φ for K epochs
end for
```

F00 is considered since they share a common OD pair and close schedule. However, the scheduled departure time of F00, 05:38, is nine minutes earlier than the scheduled departure time of F09, 05:47. Despite an incurred mild delay of F00, the optimal solution decides to hold the departure time of F00 for nine minutes in order to accommodate the passengers from F09. Moreover, since T02 has a larger seat capacity than T00, the flight F00 can accommodate more passengers if it is assigned to T02, which accounts for the T00–T02 swap action. As for the situation of F05 which also experiences disruption, the only candidate flight for rerouting is F02 since they share a common OD pair (ATL, ORD). However, F02 arrives even later than the disrupted F05, thus no passenger is rerouted from F02 to F05 and F05 remains 83 passengers. To mitigate the propagated delay as much as possible, the departure

Algorithm 4 Schedule Generator

Input: Airports dataset, aircraft dataset, # of aircrafts, operational parameters
Output: Flight schedule

for each $u \in U^{ac}$ **do**
  $u.flts \leftarrow$ empty list
  while True **do**
    $n \leftarrow u.flts.length$
    $f \leftarrow$ create new flight object
    if $n == 0$ **then**
      $f.Ori \leftarrow$ sample an airport from the airports set weighted by airport flow volume
      $f.SDT \leftarrow u.EDT + u.BriefTime$
    else
      $f.Ori \leftarrow u.flts[n].Des$
      $f.CT \sim U(u.MinCT, u.MaxCT)$
      $f.SDT \leftarrow u.flts[n].SAT + f.CT$
    end if
    if $n \geq 1$ **and** a certain probability ϵ is met **then**
      $f.Des \leftarrow u.flts[n].Ori$
    else
      $neighbors \leftarrow$ the set of connectable airports of $f.Ori$ in connectable graph
      $f.Des \leftarrow$ sample an airport from $neighbors$ weighted by airport flow volume
    end if
    $f.FT \leftarrow$ calculate the flight time based on the distance between $f.Ori$ and $f.Des$
    $f.SAT \leftarrow f.SDT + f.FT$
    if $f.SAT \geq u.LAT$ **then**
      break
    end if
    $f.Pax \leftarrow u.LoadFactor * u.Cap$
    $f.Crew \leftarrow$ AssignCrew(u, f)
    $u.flts.append(f)$
    $f.tail \leftarrow u$
  end while
end for

times of F06 and F07 are set to 30 min right after the arrival times of the previous flights, which is the minimum connection time for all entities.

Appendix B. PPO training algorithm

We present the PPO training algorithm in the following. We construct and train the actor network (parameterized by θ) and the critic network (parameterized by ϕ) in a separate manner. The loss function (i.e., the negative of learning objective) of actor network is given by

$$L^{PF}(\theta) = -\hat{\mathbb{E}}_t \left[\min\left(g_t(\theta)\hat{A}_t, \text{clip}\left(g_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{B.1}$$

$$g_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \tag{B.2}$$

where $g_t(\theta)$ denotes the probability ratio between current policy and old policy (i.e., policy in the previous episode), $\hat{A}_t$ is an estimator of the advantage function at timestep t which measures the goodness of current state, and ϵ is a hyperparameter which clips the probability ratio $g_t(\theta)$ within the interval $[1-\epsilon, 1+\epsilon]$. The expectation over time of the minimum value of clipped and unclipped terms serves as a pessimistic lower bound to the loss magnitude, which leads to a smaller update to the weights of actor network. The estimator of the advantage function at each timestep t is given by

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T_s-t+1}\delta_{T_s-1} \tag{B.3}$$

$$\delta_t = r_t + \gamma V_{\phi_{old}}\left(s_{t+1}\right) - V_{\phi_{old}}\left(s_t\right) \tag{B.4}$$

where r_t is the reward function at timestep t, $V_{\phi_{old}}$ is the estimated value of state output by the critic network in the previous episode, γ is the reward discount factor and λ is a smoothing parameter which reduces the variance of $\hat{A}_t$. If the actor network and critic

Algorithm 5 Crew Assigner

```
1: class CrewAssigner
2:     attribute
3:         crew2flts ← empty dictionary (list as the datatype of values)
4:         crew2FT ← empty dictionary (float as the datatype of values)
5:     end attribute
6:     function AssignCrew(u, f)
7:         for each c ∈ crew2flts.keys do
8:             n ← crew2flts[c].length
9:             first_f, last_f ← crew2flts[c][0], crew2flts[c][n]
10:            if last_f.Des == f.Ori and c.MinCT ≤ f.SDT − last_f.SAT ≤ c.MaxCT then
11:                if crew2FT[c] + f.FT ≤ c.MaxFightTime
12:                    and f.SAT − first_f.SDT ≤ c.MaxDutyTime
13:                    and n + 1 ≤ c.MaxLandTimes then
14:                    crew2flts[c].append(f)
15:                    crew2FT[c] ← crew2FT[c] + f.FT
16:                    return c
17:                end if
18:            end if
19:        end for
20:        c′ ← create new crew team    // no available crew team found to be assigned to flight f
21:        crew2flts[c′] ← [f]
22:        return c′
23:    end function
24: end class
```

network are constructed separately (i.e., without shared structure), the general loss concerning policy function and value function is given by

$$L^{PF+VF}(\theta,\phi) = L^{PF}(\theta) + cL^{VF}(\phi) \tag{B.5}$$

$$L^{VF}(\phi) = \hat{\mathbb{E}}_t\left[(V_\phi(s_t) - V_{\phi_{old}}(s_t) - \hat{A}_t)^2\right] \tag{B.6}$$

where $L^{VF}(\phi)$ is the mean-squared-error (MSE) loss for the critic, c is a parameter (usually 0.5). The PPO-based algorithm is shown in Algorithm 3. In each episode, the actor utilizes the old policy to collect T_s time steps of data. Then the overall loss function L^{PF+VF} is optimized with respect to parameters θ and ϕ for K epochs.

Appendix C. Generation of the airline operational datasets

In the following, we provide the algorithms that are applied to generate flight schedules, crew routing and passenger itineraries. First, we generate the flight schedule according to Algorithm 4. To generate a trajectory for each aircraft, we constantly create flight objects and add them to the aircraft trajectory until the scheduled arrival time of a newly created flight is later than the latest arrival time of the aircraft. The origin/destination airport, scheduled departure/arrival time of each flight are determined in lines 7 to 21. The probability ϵ in line 14 is related to the pattern of trajectory, which refers to a hub-and-spoke pattern if it equals to 1 while a point-to-point pattern if it equals to 0; we set it to 0.3 in this study.

The procedure to assign a crew team to each flight in aircraft trajectories is reported in Algorithm 5. The CrewAssigner class has an attribute $crew2flts$ to store the flight sequence assigned to each crew team and an attribute $crew2FT$ to store the experienced flight time of each crew team. The member function AssignCrew receives an aircraft u assigned with flight f in line 26 of Algorithm 4. If there is an existing crew team c satisfying connection constraints and several constraints on crew duty, we assign this existing crew team c to the flight f, otherwise, we create a new crew team c' to handle the flight f. In line 10, the constraint of minimum and maximum sit-times (i.e., $MinCT$ and $MaxCT$) between consecutive flights should be satisfied along with the spatial constraint of airports. The constraints on duty includes maximum total flying time (i.e., the total number of hours of actual flying time, line 11), maximum duty time (i.e., the flying times and the sit-times in the duty, line 12) and maximum number of landings (line 13).

Based on the generated flight schedule, we further generate the passenger itinerary table in Algorithm 6. The main idea to generate practical passenger itineraries in this algorithm is to distribute the passengers in each flight to direct/two-hop/three-hop itineraries based on different preset fractions, denoted by η_1, η_2, η_3 ($\eta_1 + \eta_2 + \eta_3 = 1$). Itineraries with more than three hops are not considered for simplicity and practicality. If there is an existing itinerary it satisfying spatial–temporal constraints in line 8, a fraction of passengers of the current flight f will open a new itinerary new_it that joins the existing itinerary it with the current flight leg f, which may take a length of two or three accordingly. The spatial constaint refers to the matching of OD airports in

Algorithm 6 Itinerary Assigner

Input: Set of flight objects F, fraction of direct itinerary η_1 and two-hop itinerary η_2
Output: Passenger itinerary table

$itin2flts \leftarrow$ empty dictionary (list as the datatype of values)
$itin2pax \leftarrow$ empty dictionary (integer as the datatype of values)
for each $f \in F$ **do**
  $leave\_sum \leftarrow 0$
  for each $it \in itin2flts.keys$ **do**
    $n \leftarrow itin2flts[it].length$
    $last\_f \leftarrow itin2flts[it][n]$
    if $last\_f.Des == f.Ori$ **and** $it.MinCT \leq f.SDT - last\_f.SAT \leq it.MaxCT$ **then**
      $new\_it \leftarrow$ create new itinerary
      $flag \leftarrow False$
      if $n == 1$ **then**
        $leave \leftarrow \text{int}(\eta_2 * f.Pax)$
        $itin2flts[new\_it] \leftarrow [last\_f, f]$
        $flag \leftarrow True$
      else if $n == 2$ **then**
        $leave \leftarrow \text{int}((1 - \eta_1 - \eta_2) * f.Pax)$
        $itin2flts[new\_it] \leftarrow itin2flts[it] + [f]$
        $flag \leftarrow True$
      end if
      if $flag$ **then**
        $itin2pax[new\_it] \leftarrow leave$
        $itin2pax[it] \leftarrow itin2pax[it] - leave$
        $leave\_sum \leftarrow leave\_sum + leave$
      end if
    end if
  end for
  $local\_it \leftarrow$ create new itinerary // corresponds to the itinerary that takes direct flight f
  $itin2flts[local\_it] \leftarrow [f]$
  $itin2pax[local\_it] \leftarrow f.Pax - leave\_sum$
end for
$ItineraryTable \leftarrow$ generate passenger itinerary table based on $itin2flts$ and $itin2pax$
return $ItineraryTable$

consecutive flights, while the temporal constraint ensures that the passenger transfer time between consecutive flights is within the practical range. Then, the number of passengers in it and new_it is updated and the number of passengers that transfer to new_it will be added to $leave\_sum$. The difference between the number of passengers in flight f and $f.Pax$ and $leave\_sum$ will be the number of passengers in the direct itinerary of flight f. We perform the aforementioned procedure for all the scheduled flights and finally obtain the passenger itinerary table based on dictionaries $itin2flts$ and $itin2pax$.

References

Achiam, J., 2018. Spinning up in deep reinforcement learning.
Aktürk, M.S., Atamtürk, A., Gürel, S., 2014. Aircraft rescheduling with cruise speed control. Oper. Res. 62 (4), 829–845.
Alcaraz, J.J., Losilla, F., Caballero-Arnaldos, L., 2022. Online model-based reinforcement learning for decision-making in long distance routes. Transp. Res. E Logist. Transp. Rev. 164, 102790.
Arıkan, U., 2014. Airline Disruption Management. Middle East Technical University.
Arıkan, U., Gürel, S., Aktürk, M.S., 2017. Flight network-based approach for integrated airline recovery with cruise speed control. Transp. Sci. 51 (4), 1259–1287.
Ball, M., Barnhart, C., Dresner, M., Hansen, M., Neels, K., Odoni, A., Peterson, E., Sherry, L., Trani, A., Zou, B., Britto, R., Fearing, D., Swaroop, P., Uman, N., Vaze, V., Voltes, A., 2010a. Total delay impact study: A comprehensive assessment of the costs and impacts of flight delay in the United States.
Ball, M., Barnhart, C., Dresner, M., Hansen, M., Neels, K., Odoni, A., Peterson, E., Sherry, L., Trani, A., Zou, B., et al., 2010b. Total delay impact study. In: NEXTOR Research Symposium, Washington DC.
Barnhart, C., 2009. Irregular operations: Schedule recovery and robustness. In: The Global Airline Industry. Wiley Online Library, pp. 253–274.
Barnhart, C., Boland, N.L., Clarke, L.W., Johnson, E.L., Nemhauser, G.L., Shenoi, R.G., 1998. Flight string models for aircraft fleeting and routing. Transp. Sci. 32 (3), 208–220.
Barnhart, C., Fearing, D., Vaze, V., 2014. Modeling passenger travel and delays in the national air transportation system. Oper. Res. 62 (3), 580–601.
Basso, R., Kulcsár, B., Sanchez-Diaz, I., Qu, X., 2022. Dynamic stochastic electric vehicle routing with safe reinforcement learning. Transp. Res. E Logist. Transp. Rev. 157, 102496.
Bongiovanni, C., Kaspi, M., Cordeau, J.-F., Geroliminis, N., 2022. A machine learning-driven two-phase metaheuristic for autonomous ridesharing operations. Transp. Res. E Logist. Transp. Rev. 165, 102835.

Bratu, S., Barnhart, C., 2006. Flight operations recovery: New approaches considering passenger recovery. J. Sched. 9, 279–298.
Cadarso, L., Vaze, V., 2021. Passenger-centric integrated airline schedule and aircraft recovery. Available at SSRN 3814705.
Campbell, J.F., O'Kelly, M.E., 2012. Twenty-five years of hub location research. Transp. Sci. 46 (2), 153–169.
Clausen, J., Larsen, A., Larsen, J., Rezanova, N.J., 2010. Disruption management in the airline industry—Concepts, models and methods. Comput. Oper. Res. 37 (5), 809–821.
Dennis, N., 2007. End of the free lunch? The responses of traditional European airlines to the low-cost carrier threat. J. Air Transp. Manag. 13 (5), 311–321.
Eggenberg, N., Salani, M., Bierlaire, M., 2010. Constraint-specific recovery network for solving airline recovery problems. Comput. Oper. Res. 37 (6), 1014–1026.
Federal Aviation Administration, U.S.D.o.T., 2018. Aircraft characteristics database.
Filom, S., Amiri, A.M., Razavi, S., 2022. Applications of machine learning methods in port operations–A systematic literature review. Transp. Res. E Logist. Transp. Rev. 161, 102722.
Gürkan, H., Gürel, S., Aktürk, M.S., 2016. An integrated approach for airline scheduling, aircraft fleeting and routing with cruise speed control. Transp. Res. C 68, 38–57.
Haouari, M., Zeghal Mansour, F., Sherali, H.D., 2019. A new compact formulation for the daily crew pairing problem. Transp. Sci. 53 (3), 811–828. http://dx.doi.org/10.1287/trsc.2018.0860.
Hassan, L., Santos, B.F., Vink, J., 2021. Airline disruption management: A literature review and practical challenges. Comput. Oper. Res. 127, 105137.
Herrema, F., Curran, R., Hartjes, S., Ellejmi, M., Bancroft, S., Schultz, M., 2019. A machine learning model to predict runway exit at Vienna airport. Transp. Res. E Logist. Transp. Rev. 131, 329–342.
Hu, Y., Miao, X., Zhang, J., Liu, J., Pan, E., 2021. Reinforcement learning-driven maintenance strategy: A novel solution for long-term aircraft maintenance decision optimization. Comput. Ind. Eng. 153, 107056.
Hu, Y., Song, Y., Zhao, K., Xu, B., 2016. Integrated recovery of aircraft and passengers after airline operation disruption based on a GRASP algorithm. Transp. Res. E Logist. Transp. Rev. 87, 97–112.
Khan, W.A., Ma, H.-L., Ouyang, X., Mo, D.Y., 2021. Prediction of aircraft trajectory and the associated fuel consumption using covariance bidirectional extreme learning machines. Transp. Res. E Logist. Transp. Rev. 145, 102189.
Kohl, N., Larsen, A., Larsen, J., Ross, A., Tiourine, S., 2007. Airline disruption management—perspectives, experiences and outlook. J. Air Transp. Manag. 13 (3), 149–162.
Kravaris, T., Lentzos, K., Santipantakis, G., Vouros, G.A., Andrienko, G., Andrienko, N., Crook, I., Garcia, J.M.C., Martinez, E.I., 2022. Explaining deep reinforcement learning decisions in complex multiagent settings: towards enabling automation in air traffic flow management. Appl. Intell. 1–36.
Lee, J., Lee, K., Moon, I., 2022. A reinforcement learning approach for multi-fleet aircraft recovery under airline disruption. Appl. Soft Comput. 129, 109556.
Lee, J., Marla, L., Jacquillat, A., 2020. Dynamic disruption management in airline networks under airport operating uncertainty. Transp. Sci. 54 (4), 973–997.
Liang, Z., Xiao, F., Qian, X., Zhou, L., Jin, X., Lu, X., Karichery, S., 2018. A column generation-based heuristic for aircraft recovery problem with airport capacity constraints and maintenance flexibility. Transp. Res. B 113, 70–90.
Marla, L., Vaaben, B., Barnhart, C., 2017. Integrated disruption management and flight planning to trade off delays and fuel burn. Transp. Sci. 51 (1), 88–111.
Marzuoli, A., Boidot, E., Colomar, P., Guerpillon, M., Feron, E., Bayen, A., Hansen, M., 2016. Improving disruption management with multimodal collaborative decision-making: A case study of the Asiana crash and lessons learned. IEEE Trans. Intell. Transp. Syst. 17 (10), 2699–2717.
O'Kelly, M.E., 2015. Network hub structure and resilience. Netw. Spat. Econ. 15 (2), 235–251.
O'Kelly, M.E., Bryan, D., 1998. Hub location with flow economies of scale. Transp. Res. B 32 (8), 605–616.
OurAirports, 2022. OurAirports database.
Petersen, J.D., Sölveling, G., Clarke, J.-P., Johnson, E.L., Shebalov, S., 2012. An optimization approach to airline integrated recovery. Transp. Sci. 46 (4), 482–500.
Pham, D.-T., Tran, P.N., Alam, S., Duong, V., Delahaye, D., 2022. Deep reinforcement learning based path stretch vector resolution in dense traffic with uncertainties. Transp. Res. C Emerg. Technol. 135, 103463.
Rapajic, J., 2018. Beyond Airline Disruptions: Thinking and Managing Anew. Routledge.
Ruan, J., Wang, Z., Chan, F.T., Patnaik, S., Tiwari, M.K., 2021. A reinforcement learning-based algorithm for the aircraft maintenance routing problem. Expert Syst. Appl. 169, 114399.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
Šemrov, D., Marsetič, R., Žura, M., Todorovski, L., Srdic, A., 2016. Reinforcement learning approach for train rescheduling on a single-track railway. Transp. Res. B 86, 250–267.
Serrano, F.J.J., Kazda, A., 2017. Airline disruption management: Yesterday, Today and Tomorrow. Transp. Res. Procedia 28, 3–10.
Sinclair, K., Cordeau, J.-F., Laporte, G., 2016. A column generation post-optimization heuristic for the integrated aircraft and passenger recovery problem. Comput. Oper. Res. 65, 42–52.
Teodorović, D., Guberinić, S., 1984. Optimal dispatching strategy on an airline network after a schedule perturbation. European J. Oper. Res. 15 (2), 178–182.
Teodorović, D., Stojković, G., 1995. Model to reduce airline schedule disturbances. J. Transp. Eng. 121 (4), 324–331.
Thengvall, B.G., Bard, J.F., Yu, G., 2003. A bundle algorithm approach for the aircraft schedule recovery problem during hub closures. Transp. Sci. 37 (4), 392–407.
Wang, K., Jacquillat, A., 2020. A stochastic integer programming approach to air traffic scheduling and operations. Oper. Res. 68 (5), 1375–1402.
Wen, X., Chung, S.-H., Ji, P., Sheu, J.-B., 2022. Individual scheduling approach for multi-class airline cabin crew with manpower requirement heterogeneity. Transp. Res. E Logist. Transp. Rev. 163, 102763.
Wen, X., Ma, H.-L., Chung, S.-H., Khan, W.A., 2020. Robust airline crew scheduling with flight flying time variability. Transp. Res. E Logist. Transp. Rev. 144, 102132.
Wen, X., Sun, X., Sun, Y., Yue, X., 2021. Airline crew scheduling: Models and algorithms. Transp. Res. E Logist. Transp. Rev. 149, 102304.
West, J., Ventura, D., Warnick, S., 2007. Spring Research Presentation: a Theoretical Foundation for Inductive Transfer. Vol. 1. No. 08. Brigham Young University, College of Physical and Mathematical Sciences.
Xiong, J., Hansen, M., 2013. Modelling airline flight cancellation decisions. Transp. Res. E Logist. Transp. Rev. 56, 64–80.
Xu, Y., Wandelt, S., Sun, X., 2021. Airline integrated robust scheduling with a variable neighborhood search based heuristic. Transp. Res. B 149, 181–203.
Yan, Y., Chow, A.H., Ho, C.P., Kuo, Y.-H., Wu, Q., Ying, C., 2022. Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities. Transp. Res. E Logist. Transp. Rev. 162, 102712.
Zuidberg, J., 2014. Identifying airline cost economies: An econometric analysis of the factors affecting aircraft operating costs. J. Air Transp. Manag. 40, 86–95.