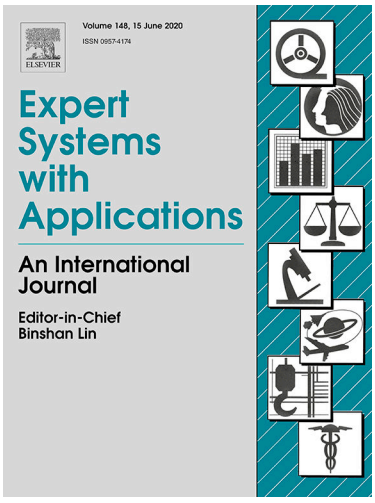

A reinforcement learning-based algorithm for the aircraft maintenance routing problem

J.H. Ruan, Z.X. Wang, Felix T.S. Chan, S. Patnaik, M.K. Tiwari

PII: S0957-4174(20)31071-X
DOI: https://doi.org/10.1016/j.eswa.2020.114399
Reference: ESWA 114399

To appear in: *Expert Systems with Applications*

Received Date: 11 May 2019
Revised Date: 4 November 2020
Accepted Date: 27 November 2020

Please cite this article as: Ruan, J.H., Wang, Z.X., Chan, F.T.S., Patnaik, S., Tiwari, M.K., A reinforcement learning-based algorithm for the aircraft maintenance routing problem, *Expert Systems with Applications* (2020), doi: https://doi.org/10.1016/j.eswa.2020.114399

A reinforcement learning-based algorithm for the aircraft maintenance routing problem

J.H. Ruan[a,b,c], Z.X. Wang[d,c*], Felix T.S. Chan[b], S. Patnaik[b,e] and M.K. Tiwari[f]

[a]College of Economics and Management, Northwest A & F University, Yangling, China.
[b]Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hom, Hong Kong.
[c]Institute of Systems Engineering, Dalian University of Technology, Dalian, China.
[d]School of Business Administration, Dongbei University of Finance and Economics, Dalian, China.
[e]Eepartment of Industrial and Systems Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India.
[f] National Institute of Industrial Engineering(NITIE) Vhar Lake Mumbai 400087, India.

Email: rjh@nwafu.edu.cn; wangzhengxu@dufe.edu.cn; f.chan@polyu.edu.hk; patanaiksovit@gmail.com; mkt09@hotmail.com

*Corresponding author: Z.X. Wang; Email: wangzhengxu@dufe.edu.cn

Abstract

With recent developments in the airline industry worldwide, the competition among the industry has increased largely with many key players in the market. In order to generate profits, the industry has paid much attention to generate optimal routes that are maintenance feasible. The main aim of operational aircraft maintenance routing problem (OAMRP) is to generate these optimal routes for each aircraft that are maintenance feasible and follow the constraints defined by the Federal Aviation Administration (FAA). In this paper, the OAMRP is studied with two main objectives. First, to propose a formulation of a network flow-based Integer Linear Programming (ILP) framework for the OAMRP that considers three main maintenance constraints simultaneously: maximum flying-hour, limit on the number of take-offs between two consecutive maintenance checks and the work-force capacity. Second, to develop a new reinforcement learning-based algorithm which can be used to solve the problem, quickly and efficiently, as compared to commonly available optimization software. Finally, the evaluation of the proposed algorithm on real case datasets obtained from a major airline located in the Middle East verifies that the algorithm generates high-quality solutions quickly for both medium and large-scale flight schedule dataset.

Keywords: Aircraft routing problem, Sequential decision-making problem, Markov Decision Process (MDP), Reinforcement Learning,

1. Introduction

The air transportation system is one of the most complex man-made transportation system in the world. The aviation industry has developed to a large extent in the past decade and contributes more to the global GDP than other industries or sectors (Ball et al., 2007). As a result of this development, the competition among the airlines has increased to a large extent due to which the airlines had to cut their ticket prices down; this finally led to a large increase in demand among the passengers.

The planning and scheduling of aircrafts are one of the most time-sensitive and critical function in the airline industry and operations research has played a major role in helping the industry in this sector. The airline operations problem can be divided into four major sub-problems, i.e. a. Flight Scheduling, b. Fleet Assignment, c. Aircraft Maintenance Routing and d. Crew Management. These sequential operations are performed before the departure of an aircraft. First, the flight schedule is done based on corporate strategic plan and marketing

issues such as demand among the passengers (Lee et al., 2007; Yan et al., 2007). This is done by the airline in order to decide which cities to cover, at what times and at which frequency. Second, the fleet assignment problem is solved where the airlines assign aircraft types to the scheduled flight legs based on their maximum total range, number of seats, operating costs etc. After this, a sequence of flight legs to be flown by each aircraft will be determined so that each flight leg is covered exactly once. This process is known as aircraft maintenance routing, and maintenance constraints play a major role in deciding the routes of each aircraft as because each aircraft has to undergo maintenance checks after a proper interval of time. Finally, the crew management is done where crew members are assigned to each aircraft keeping in mind their duty cost and collective agreements. In this paper, the operational aircraft maintenance routing problem (OAMRP) of the airlines is studied with the assumption that both the previous stages are already solved before, i.e. Flight Scheduling and Fleet Assignment.

As mandated by the Federal Aviation Administration (FAA), there are some upper limit restrictions for the cumulative flying hours, number of consecutive flight days and the total number of take-off for an aircraft. Beyond this limit, a number of predetermined maintenance checks of the aircraft must be done by the airline. There are four types of maintenance checks, as defined by the FAA, which have different durations and frequencies depending on their type (Clarke et al., 1997). The *Type-A* maintenance check is conducted for every 65 flight-hours or once every four days, and it includes visual inspection of major systems such as landing gears, engine and control surfaces. This check lasts for about 4-8 hours. Each aircraft undergoes the *Type-B* maintenance check once for every 300-600 flight-hours where a thorough inspection and lubrication of all moving parts are done. The *Type B* check lasts for about 1-3 days. *Type C* and *Type D* checks a performed once every one to four years where the aircraft is taken out of service for about one month (Sriram & Haghani, 2003).

In this paper, a network flow-based Integer Linear Programming (ILP) framework is formulated for the OAMRP that considers three important maintenance constraints simultaneously: maximum flying-hour, limit on the number of take-offs between two consecutive maintenance checks and the work-force capacity. The OAMRP is then shown to be modelled as a sequential decision-making problem using Markov Decision Process (MDP), which is solved by using existing reinforcement learning-based method in order to generate optimal routes that are maintenance feasible. The main motive of the paper is to incorporate the existing reinforcement learning-based algorithm to the field of aircraft

routing. The remainder of the paper is structured as follows. Section 3 discusses the contribution and motivation of the paper in details. Section 4 presents the ILP formulation for the proposed OAMRP. Section 5 gives a background knowledge about the mathematics behind reinforcement learning. In Section 6, the proposed reinforcement learning-based algorithm is explained in detail. Experimental results and model analysis are shown in Section 7. Finally, Section 8 of the paper summarizes the conclusion and outlines the direction for future work.

2. Literature review

In the past two decades, there has been extensive research in the field of aircraft scheduling, especially on the aircraft maintenance routing problem (AMRP), and extensive number of surveys have been published (Eltoukhy, Chan, & Chung, 2017; Etschmaier & Mathaisel, 1985; Gopalakrishnan & Johnson, 2005; Sherali, Bish, & Zhu, 2006). In most of the literature, AMRP was studied on two major aspects, i.e. tactical aspect and operational aspect. Generally, the earlier studies on AMRP addressed the tactical side of the problem where specific rotation of flights were generated for each aircraft without considering the maintenance constraints and other dynamic issues. In reality, following one single rotation of flight legs for each and every aircraft may not be possible because of operational maintenance constraints. However, on the operational side, it takes into account all real-life operational maintenance constraints while generating routes for each aircraft that are maintenance feasible routes. As in Haouari et al. (2012), there are three important maintenance constraints that must be followed while solving the AMRP, these are as follows: a) each aircraft must undergo one maintenance check every 4 days, b) the total flying hour of each aircraft between two consecutive maintenance checks must not exceed the maximum allowable flying hour limit, c) similarly the total number of take-offs of each aircraft must not go beyond the allowed maximum allowable number of take-offs.

Among the earlier studies on the tactical aspect of the maintenance routing problem, a set-partitioning based model of AMRP was developed by Kabbani & Patty (1992) where they tried to solve the three-day maintenance routing problem with the assumption that maintenance checks are carried overnight. They determined the maintenance viable routes also known as the line of flights (LOF) for each aircraft using a two-step heuristic method where first they constructed over-the-day routes and then connected them to construct the tours. Gopalan and Talluri (1998) expanded the use of LOF in order to study the three-day

Routing problem. They proposed a simple polynomial time model for solving the static (fixed routes) and dynamic (routes are not fixed) formulation. A string-based model with aircraft utilization constraints was presented by Barnhart et al. (1998) for the AMRP with the objective to minimize the total cost of selected strings. Here a string refers to maintenance viable routes of an aircraft. The proposed model was solved using the branch-and-price approach. As shown in the studies of Mak & Boland (2000), AMRP can also be modelled as an asymmetric travelling-salesman problem. They used Lagrangian Relaxation (LR) to find the lower bounds and a heuristic approach that used simulated annealing (SA) to find the upper bounds. Recently, a novel rotation-tour network model (RTNM) based on a modified time-space network was formulated for the maintenance routing problem by Liang et al., 2011, and a new mixed integer linear program (MILP) framework was proposed in accordance to the network. The developed framework was solved using CPLEX 10.0 software (IBM ILOG CPLEX Optimization Studio, often informally referred to simply as CPLEX, is a commercial optimization software package). Their main objective was to minimize the connection cost and maintenance cost and assumed that maintenance was carried out overnight. The number of passengers who remain on the same aircraft and do not change the aircraft determines the through value between two connected flight legs.

In contrast to the above-mentioned tactical methods, many studies have also been conducted on the operational side of the AMRP (OAMRP). Sriram & Haghani (2003) created an integer linear programming model and took into consideration two of the above-discussed maintenance constraints, that is, one maintenance check for every four days and work-force capacity. They adopted a notation similar to the LOFs, known as Origin-Destination pairs (OD) and assumed that maintenance checks were carried during the night. The authors also formulated an effective heuristic algorithm for solving the same in reasonable time as compared to the time taken by CPLEX software. CPLEX developed by IBM is one of the most commonly used commercial software for solving optimization problems. They also extended their framework to consider the total flying hour constraint of each aircraft but didn't solve it due to its high complexity. The OAMRP based on daily routing basis was considered instead of long-term routing by Sarac et al. (2006). They proposed a set partitioning based formulation to minimise the total number of unused legal flying hours. On the operational side, they took into account only the allowable maximum flying-hour constraint and the work-force capacity constraint. Since the OAMRP was modelled using set-portioning based formulation, it produced exponential number of feasible routes; as a result,

they proposed a branch-and-price method to solve the model. Branch-and-price method is generally implemented for very large-scale integer programs (Barnhart et al. 1998). Similarly, Haouari et al. (2012) developed a polynomial sized nonlinear model for the same problem. They used the reformulation-linearization based technique to linearize the non-linear model and obtained good quality results. An Integer Linear Programming (ILP) based method was proposed by Orhan et al. (2011) for maximising the utilization of the remaining time for a given maintenance schedules. More recently, Al-Thani et al. (2016) proposed a very-large scale neighbourhood search algorithm (VLNS) for the operational aircraft maintenance routing problem where they tried to minimize the sum of the remaining flying times. In order to validate their model they used a dataset of flights including 354 legs and were scheduled to be flown by an eight-aircraft fleet during a one week planning horizon. Whereas, here it is worth noting that, in our paper very large scale datasets are used in-order to validate the proposed model, i.e. datasets having up to 400 flight legs scheduled to be flown by 42 aircraft fleet over a 4-days planning horizon. Eltoukhy et al. (2017) formulated an ILP based model for the operational aircraft maintenance routing problem using two important maintenance constraints i.e. limit on cumulative flying hour of each aircraft and workforce capacity. For the first time they proposed to solve the model using heuristic methods based on Genetic Algorithm (GA), Simulated Annealing (SA), Ant Colony Optimization (ACO). They extended their model in Eltoukhy et al. (2018) and proposed a heuristic based solution algorithm that efficiently solves the OAMRP within a much smaller amount of time. Table 1 shows in details about the above AMRP models. Safaei & Jardine (2018) formulated an ILP based framework for the OAMRP with the aim to minimize the number of maintenance misalignments (the gap between the available maintenance capacity and demand). They proposed a new solution methodology based on local search for solving the ILP frame work in order to obtain optimal solution. Most recently, Sanchez et al. (2020) proposed a multi-objective mixed integer linear programming (MMILP) formulation for the OAMRP in order to minimize the number of maintenance regulation violations and the number of not airworthy aircraft. They proposed a novel heuristic based iterative solution procedure for solving the MMILP in order to obtain optimal results. In order to make this paper more applicable to the real world problem, three operational constraints are used namely M2 (The total flying hour of each aircraft between two consecutive maintenance checks must not go beyond the maximum allowable flying hour limit), M3 (The total number of take-offs of each aircraft must not go beyond the allowed maximum allowable number of take-offs limit), M4

(The number of aircrafts to be maintained at each maintenance station should not exceed the available workforce capacity).

Table 1. Summary of AMRP literature

Authors / Year	Planning Horizon	Network	Model	Objective	Solution method	Operational Constrains
Kabbani & Patty (1992)	3 Days	-	Set Partitioning	Minimize total cost	Heuristic	-
Barnhart et al. (1998)	Weekly	Connection Network	Set Partitioning	Minimize total cost	Branch and price	-
Gopalan and Talluri (1998)	3 Days	Connection Network	Euler Tour	Find feasible routes	Polynomial time model	-
Mak & Boland (2000)	-	Connection Network	Asymmetric Travelling-Salesman	Maximize utilization of remaining flying time	Lagrangian-Relaxation, SA	-
Sriram & Haghani (2003)	Weekly	Connection Network	Integer Linear Programming	Minimize maintenance & re-assignment cost	Depth-first search & random search	M1, M4
Sarac et al. (2006)	Daily	Connection Network	Set Partitioning	Minimize number of unused legal flying hour	Branch-and-price	M2, M4
Liang et al. (2011)	Daily	Time-space Network	Network Flow	Minimize connection & maintenance cost	CPLEX 10.0	-
Haouari et al. (2012)	Daily	Connection Network	Nonlinear Programming	Find feasible routes	CPLEX 12.1	M1, M2, M3
Al-Thani et al. (2016)	Weekly	Connection Network	Mixed-Integer Programming	Minimize the sum of remaining flying time	Very-large scale neighbourhood search algorithm	M1, M2, M3
Eltoukhy et al. (2017)	4 Days	Connection Network	Integer Linear Programming	Maximize the through value	ACO, GA, SA	M2, M4
Safaei et al. (2018)	Weekly	Connection Network	Integer Linear Programming	Minimize the number of maintenance misalignments	Branch-and-bound, Local search	M1, M2, M4
Sanchez et al. (2018)	30 Days	-	Multi-objective Mixed Integer Linear Programming	Minimise maintenance violence & not airworthy aircraft	Heuristic based iterative methodology	M1, M2

M1: Each aircraft must undergo one maintenance check for every 4 days

M2: The total flying hour of each aircraft between two consecutive maintenance checks must not go beyond the maximum allowable flying hour limit

M3: The total number of take-offs of each aircraft must not go beyond the allowed maximum allowable number of take-offs limit

M4: The number of aircrafts to be maintained at each maintenance station should not exceed the available workforce capacity

3. Contribution and motivation of the paper

In this paper, we extend the OAMRP model presented in Eltoukhy et al., 2017 along with considering the constraint on the total cumulative take-offs between two successive maintenance checks and to formulate it using both ILP framework and a novel Markov Decision Process (MDP)-based framework. The ILP based framework is solved using CPLEX software and the proposed MDP framework for the OAMRP is solved based on model-free reinforcement learning method (Sutton & Barto, 2018). Finally the solution and the time taken by reinforcement learning method is compared with that of CPLEX software and other available methods in the literature. The development in reinforcement learning has led to its application in many decision support systems. In naïve reinforcement learning is, learning what to do - how to map current situations to actions in order to maximise the reward. The learner or the agent is not told about which action to take at a current situation, but instead, it must decide which action to take for a maximum reward from its past experience with the environment. The main goal of reinforcement learning is to learn optimal policy (series of state-action pairs executed by the agent) that maximises the expectation of the cumulative rewards.

Since 1996, when the work of Kaelbling et al. on reinforcement learning came into light, it has been actively used in solving many real-world decision-making problems (Aydin & Öztemel, 2000; Wang & Usher 2005; Gabel & Riedmiller, 2012; Zhang et al., 2013; Šemrov et al., 2016; Kara & Dogan, 2018). One of the few earliest work of reinforcement learning for scheduling problem is that of Aydin and Öztemel (2000). They used reinforcement learning-based agents to select suitable dispatching rules for scheduling in real time according to the shop floor conditions. Wang and Usher 2005 solved a similar type of problem, where they studied the effect of reinforcement learning algorithm on selection of dispatching rule for a single machine. Gabel and Riedmiller 2011 modelled the job-shop scheduling problem as sequential decision problem using Markov Decision Process (MDP) framework and solved it using reinforcement learning. They showed that the agents started with no prior knowledge and made poor dispatching decisions initially, but gradually improved their dispatching behaviour with time. A recent work by Šemrov et al. 2016 seems to be the first to incorporate reinforcement learning algorithm for railway scheduling. Moreover, the work focuses on recovery and minimisation of the total initial delay of trains. A few recent works have also been done in the field of inventory management which incorporates reinforcement learning in order to minimize the cost of retailers. Kara & Dogan (2018), investigated the inventory

management problem for generating optimal replenishment ordering policy of perishable products in order to minimize the total cost of a retailer. Most recently, Bharti et al. (2020) studied a similar type of problem where they applied reinforcement learning to the ordering management problem of supply chain with the objective to minimize inventory costs. They showed that reinforcement learning led to the convergence of the total cost to an optimal value and performed better as compare to two commonly used heuristic approaches.

Many recent studies have incorporated reinforcement learning in broadly related areas such as board games, playing cards and autonomous driving. For example, AphaGo (Silver et al. 2016) developed by Google DeepMind based on deep reinforcement learning (DRN) defeated the world champion Lee Sedol in the board game Go in the year 2016. In the subsequent year an improvised model of AlphaGo was developed namely AlphGo Zero, that defeated the previous AlphaGo sytem by 100-0 (Silver et al. 2017). Most recently, Chen et al. (2019) applied reinforcement learning in the field of autonomus driving. They designed an input represenation of the from view to reduce the complexity by using visula encodeing and later used DRN into the framework for autonomus driving. Recent works on integration of reinforcement learning to the field of autonomus driving has been compiled by Kiran et al. (2020). Reinforcement learning is much suitable to problems involving interacting agents that require dynamic decision.

The main motivation of using reinforcement learning for solving the operational aircraft maintenance routing problem is as follows:- Firstly, according to Gosavi (2009), reinforcement learning is a powerful tool for solving complex sequential decision-making problems, and has been efficiently used in several transportation problems. For example, traffic signal (Khamis & Gomaa, 2014), train rescheduling (Šemrov et al., 2016) , route planning (Zolfpour-Arokhlo et al., 2014) and traffic flow control (Walraven, Spaan, & Bakker, 2016; Zhu & Ukkusuri, 2015). As per the experimental results of the research efforts mentioned above, reinforcement learning is at least equivalent and often superior to traditional operation research method in terms of overall decision making performance and computational time. Secondly, the heuristic solution methods presented in the literature, keep on iterating the algorithm randomly on the feasible solution space until the best solution is achieved or the number of iteration reaches the allowable maximum number of iterations. Whereas, in reinforcement learning the agents keeps on learning from each iteration and in return, it tries to improve the quality of the next iteration based on the results of the previous one. As a result, this leads to a faster convergence of the solution. Thirdly, reinforcement

learning provides a formalism for measuring the utility of actions that gives no immediate benefits, but do give better benefits in the future (Rennie & McCallum, 1999). Fourthly, as mathematically proved by Sutton & Barto (2018), reinforcement learning algorithm based on policy improvement theorem always assures that the overall process converges to the optimal solution, given sample of episodes and no other knowledge of the environment dynamics. Finally, to the best of our knowledge, there has been no such research or document like this in the literature that uses reinforcement learning for solving the operational aircraft maintenance routing problem. Moreover, we hope that this research effort serves as a reference document for the future considerations of using reinforcement learning in solving more complex and integrated aircraft maintenance routing problem.

4. The model

For a given schedule of different flights, the main objective of the proposed model of AMRP is to determine or assign maintenance viable routes for each aircraft so as to maximize profitability. Here profitability is referred to as the total sum of the through values of routing flights through airports. A through value is the desirability of one-stop service between a pair of cities, i.e. the extra revenue that would be gained by the airline from extra passengers who are attracted by this through service (Clarke et al., 1997). In this section we extend the model presented by Eltoukhy et al. (2017) along with considering the limit on the total number of take-offs between two successive maintenance operations.

Usually, the maintenance checks considered for the maintenance routing problem are the type A checks as per the FAA. This requires each and every aircraft to undergo maintenance check for every 65 hours of cumulative flying or approximately once every four days. But airlines generally enforce different maintenance requirements that are more stringent as compared to those of FAA and require an A check for every 35-40 hours of cumulative flying. As the main objective of the paper is to move towards the real world case, three important maintenance constraints are considered in the model – a) Performing maintenance checks for every 35-40 cumulative flying hours instead of 65 hours of flying. b) Limit on the number of take-offs for each aircraft. c) Work-force availability constraint for the maintenance station in order to prevent the situation from receiving more aircraft than the available work-force capacity, as because overcrowding may lead to delay and finally cancellation of the flight

4.1 Network structure

In this paper, the AMRP formulation is presented based on the connection network as described in Haouari et al., 2012. In this type of network, the nodes represent the set of flight legs, and the arcs correspond to the viable connection between the flight legs. Generally, there exist two types of connection arcs, i.e. ordinary arcs and maintenance arcs which are represented by the decision variables X_{ij} and Y_{ij} respectively. An ordinary arc connects two flight legs i and j if the destination airport of the flight leg i is the origin airport of the flight j and also the sum of arrival time at airport i and the average Turn-Around-Time (TAT) is smaller than the flight leg j's departure time. Similarly, a maintenance arc connects two flight legs i and j such that the maintenance station is available at the arrival airport of the flight leg j. The dummy source and sink nodes are connected to the appropriate starting and the ending flight legs respectively by using the same connection arc. Note that, these dummy nodes don't represent any real-life condition, like origin airport or destination airport. These nodes are incorporated for helping the model to start and terminate the construction of routes for each aircraft (Başdere & Bilge, 2014). The schematic diagram of the flight legs and the connection between them is shown in Fig. 1.

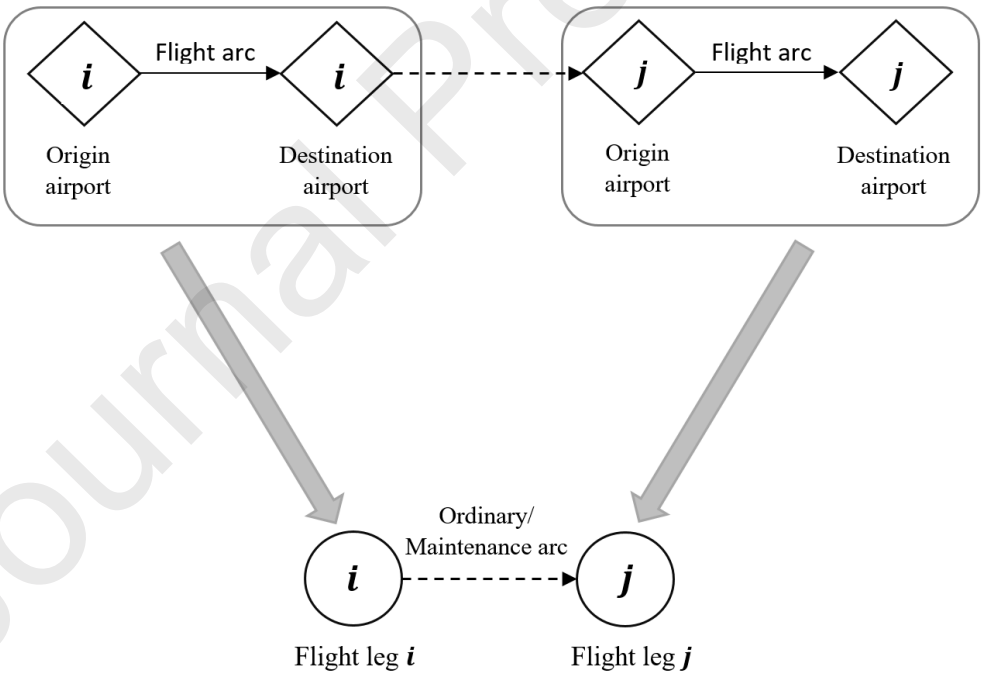

Fig. 1 Schematic diagram of the flight legs and connection between them.

4.2 Mathematical formulation

Before the detailed mathematical formulation, all the notations for the AMRP model are listed below and will be frequently used throughout the paper. First, all the sets and their respective indices are stated below.

$i, j, l \in FL$: List of flight legs' schedules.

$k \in K$: List of aircrafts.

$m \in M$: List of available maintenance stations.

FLM : List of the flight legs having maintenance station at their destination airport.

$a \in AI$: List of airports .

o : Index for dummy source node

z : Index for dummy sink node

Next, all the connection model parameters are described below:

DT_i : Flight leg i departure time, taken with respect to a reference time zone

AT_i : Flight leg i arrival time, taken with respect to the same time zone

O_{ia} : $O_{ia} = 1$ for flight leg i having origin airport as a or otherwise $O_{ia} = 0$

D_{ia} : $D_{ia} = 1$ for flight leg i having destination airport as a or otherwise $D_{ia} = 0$

FD_i : Total flying hour of leg i (duration)

C_{ij} : The through connect revenue for the through connection between leg i and leg j

T_{max} : The maximum allowable cumulative flight-hours for each aircraft between two consecutive maintenance check.

TF_{max} : The maximum allowable take-offs for each aircraft between two consecutive maintenance check.

TAT : The average time required for downed-boarding the passengers, unloading and loading of luggage, changing of gates and fuelling the aircraft

WF : Total work-force available at the maintenance station

MT : Time required for *Type A* maintenance check

L : Considerably a large number

Finally, the decision variables are as follows.

$$X_{ijk} = \begin{cases} 1 & \text{if the flight legs } i \text{ and } j \text{ are covered by the same aircraft } k \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{ijk} = \begin{cases} 1 & \text{if the aircraft } k \text{ undergoes maintenance after covering flight legs } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

Before presenting the mathematical model, the scope of the proposed operational AMRP is described as follows:

- A four-day planning horizon is assumed for the proposed model because generally, the airlines repeat their flight schedule every four days so as to satisfy the constraint of one maintenance check every four days. Moreover, the four-day planning horizon is commonly used in many works reported in the literature. Lastly, planning horizons of more than four days, for example, five or six days or a week, might produce routes that are susceptible to disruptions.
- All the maintenance visits discussed in the model only considers maintenance visit of *Type A*.
- Only the existing maintenance stations are considered in the paper, and no recommendation is done for new maintenance stations. Also, it is assumed that the maintenance stations are available only at the hub airport.
- Lastly, the workforce capacity available at each maintenance station is deterministic.

The connection network for the AMRP model is shown in Fig. 2. The main task of the paper is to find such maintenance feasible routes for each aircraft while maximizing the total through revenue. Thus, using the above-mentioned sets, parameters and decision variables the mathematical formulation of the AMRP can be written as follows:

$$\text{Maximize } \left(\sum_{k \in K} \sum_{i \in FL} \sum_{j \in FL} C_{ij}.X_{ijk} + \sum_{k \in K} \sum_{i \in FL} \sum_{j \in FLM} C_{ij}.Y_{ijk} \right) \quad (1)$$

Such that: $$\left(\sum_{k\in K}\sum_{j\in FL} X_{ijk} + \sum_{k\in K}\sum_{j\in FLM} Y_{ijk}\right) = 1 \qquad \forall\, i \in FL \tag{2}$$

$$\sum_{j\in FL} X_{ojk} + \sum_{j\in FLM} Y_{ojk} = 1 \qquad \forall\, k \in K \tag{3}$$

$$\sum_{i\in FL} X_{izk} = 1 \qquad \forall\, k \in K \tag{4}$$

$$\sum_{j\in FL\cup\{z\}} X_{ijk} + \sum_{j\in FLM} Y_{ijk} = \sum_{j\in FL\cup\{o\}} X_{jik} \qquad \forall\, i \in FL, k \in K \tag{5}$$

$$\sum_{j\in FL} X_{ojk} + \sum_{j\in FLM} Y_{ojk} = \sum_{i\in FL} X_{izk} \qquad \forall\, k \in K \tag{6}$$

$$AT_i + TAT - DT_j \le L.\left(1 - X_{ijk}\right) \qquad \forall\, i,j \in FL, k \in K \tag{7}$$

$$\sum_{k\in K} X_{ijk} \le \sum_{a\in AI} DT_{ia}O_{ja} \qquad \forall\, i,j \in FL \tag{8}$$

$$\sum_{i\in FL}\left(\sum_{j\in FL} X_{ijk} + \sum_{j\in FLM} Y_{ijk}\right) \le TF_{max} \qquad \forall\, k \in K \tag{9}$$

$$\sum_{i\in FL} FD_i.\left(\sum_{j\in FL} X_{ijk} + \sum_{j\in FLM} Y_{ijk}\right) \le T_{max} \qquad \forall\, k \in K \tag{10}$$

$$\sum_{i\in FL}\left(\sum_{j\in FLM} Y_{ijk}\right) \le WF \qquad \forall\, k \in K \tag{11}$$

$$Y_{ijk}.(AT_i + TAT) \le DT_j \qquad \forall\, i,j \in FL, k \in K \tag{12}$$

$$\sum_{k\in K} Y_{ijk} \le \sum_{a\in AI} DT_{ia}O_{ja} \qquad \forall\, i \in FL, j \in FLM \tag{13}$$

$$Y_{ijk}.(AT_i + MT - DT_j) \le L.\left(1 - X_{jlk}\right) \qquad \forall\, i,l \in FL, j \in FLM, k \in K \tag{14}$$

$$X_{ilk} + Y_{ijk} \le 1 \qquad \forall\, i,l \in FL, j \in FLM, k \in K \tag{15}$$

The objective function stated in Eq. (1) maximizes the total profit incurred from the through-connect flights (through value). This value is obtained by adding up all the feasible through connects revenues C_{ij}. The coverage constraints are described in Eq. (2) - Eq. (4). Constraints in Eq. (2) ensures that each flight leg i is assigned to only one aircraft. Eq. (3) constraints guarantees that route of each and every aircraft k starts from the source node either by utilising the ordinary arcs or the maintenance arcs. Constraint in Eq. (4) is similar to that of Eq. (3), but it guarantees that each and every aircraft k ends their respective route at the sink node.

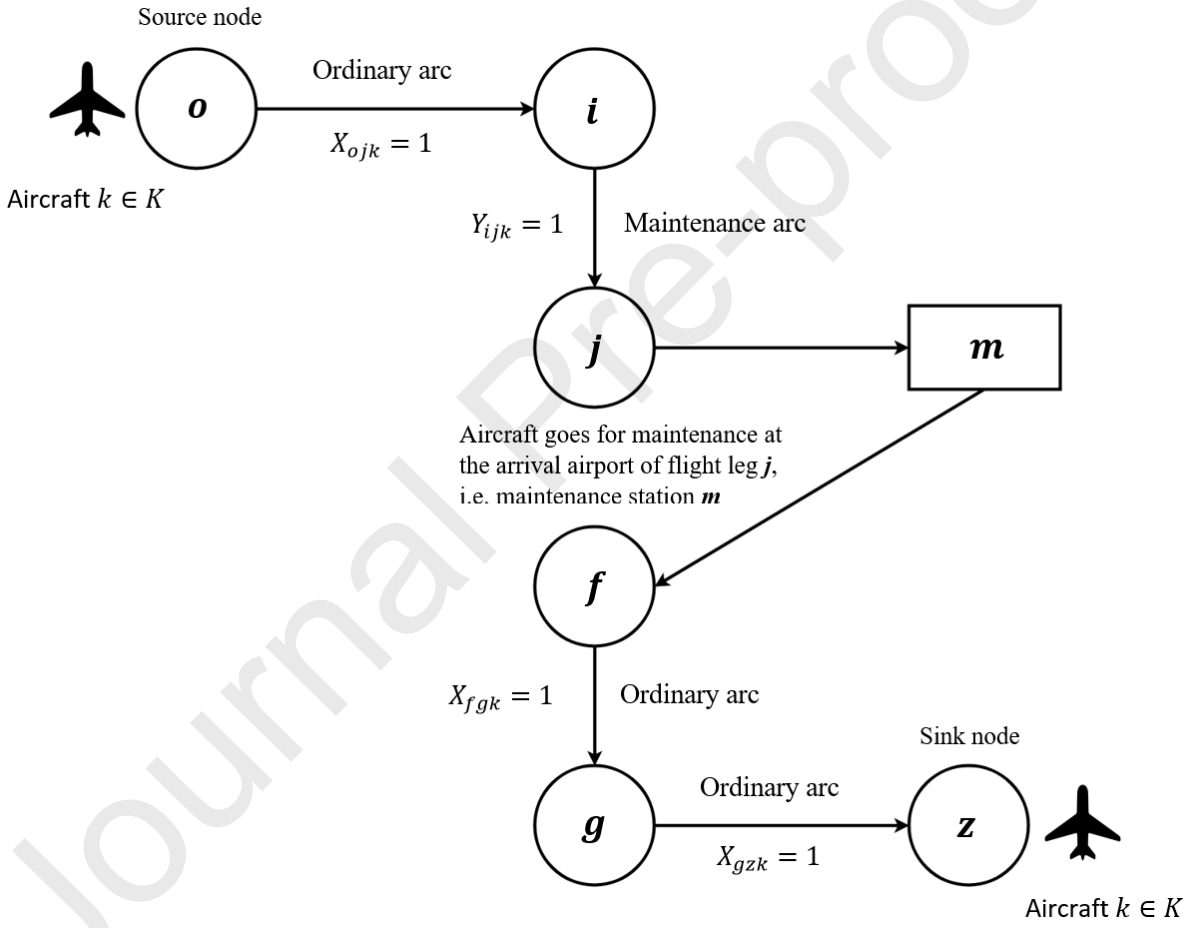

Fig. 2 Connection network representation.

The balance constraints are formulated in Eq. (5) - Eq. (6). These constraints help in maintaining the movement of aircrafts throughout the connection network. The constraints in Eq. (5) retain the balance when an aircraft k travels the nodes of the flight legs. They illustrate that if an aircraft *k* travels a flight leg *i* by using the ordinary arc, then it must travel the next flight leg *j* either by using the ordinary arcs or the maintenance arcs. The constraints in Eq. (6) ensure that all the aircrafts that start their route from the source node must end their route at the sink node so that the schedule can be repeated again.

If the same aircraft *k* connects flight leg *j* and flight leg *i*, then it needs to satisfy the time and place constraints as illustrated in Eq. (7) and Eq. (8) respectively. The time constraints in Eq. (7) states that, a flight *k* can cover flight leg *j* after covering flight leg *i*, if the sum of the turn-around time and the arrival time of the flight leg *i* is earlier than the departure time of flight leg *j*. Similarly, the place constraints in Eq. (8) indicate that the aircraft *k* can travel flight leg *j* after travelling flight leg *i*, if the destination airport of flight leg *i* is same as the origin airport of the flight leg *j*.

Until now, the above-represented balance and coverage constraints do not ensure whether an aircraft *k* should go for maintenance check or not as per FAA rules. In order to ensure that all aircrafts undergo maintenance check in the proper interval of time, the operational restrictive constraints are presented in Eq. (9) and Eq. (10). The constraints in Eq. (9) guarantee that the total number of take-offs of a particular aircraft *k* between two consecutive maintenance check does not go beyond the maximum allowable number of take-offs TF_{max}. Also the constraint in Eq. (10) ensure that the maximum cumulative flying-hours of an aircraft *k* between two consecutive maintenance operation, must be less than the allowed T_{max}. If the cumulative flying-hours goes beyond the allowable limit, then the aircraft *k* should undergo a maintenance operation. In order to assign a maintenance visit for an aircraft k, it is very necessary to ascertain whether the maintenance stations have enough work-force availability or not. These constraints are stated in Eq. (11) which ensures that the total number of aircrafts for maintenance check does not go beyond the total work-force availability. Similar to the time and place constraints in Eq. (7) and Eq. (8), it is also necessary to consider the place and time constraints for the maintenance arc as well. Hence, these constraints are illustrated in Eq. (12) - Eq. (14). The constraints in Eq. (12) are similar to those of Eq. (7), but applied during the use of the maintenance arc in place of the ordinary arc. Similar to the place

constraint for the ordinary arcs in Eq. (8), the constraints in Eq. (13) consider the place constraints for the maintenance arcs.

After completing the maintenance operation, the aircraft needs to be regulated for covering the next flight leg. In order to achieve these, the constraints in Eq. (14) were cast to guarantee that an aircraft k can travel the next subsequent flight leg j if it is ready before the departure-time of the flight leg i. The constraints in Eq. (15) ensure that each aircraft k must cover a flight leg either by using an ordinary arc or a maintenance arc.

5. Background knowledge of reinforcement learning

The reinforcement learning (RL) (Kaelbling et al., 1996; Sutton & Barto, 2018) is a computational way of learning which action to be undertaken in a given situation (state) in order to achieve one or multiple goals. RL aims at maximizing the cumulative reward collected by an agent through continuous trial-and-error interaction with the environment. This reward is necessary for the agent in order to learn its behaviour; this is also known as a reinforcement signal. The RL problem can be modelled as a Markov Decision Process (MDP) framework and can be applied for learning the optimal control policies in the MDP as described in Puterman, 2014.

In MDP the Reinforcement Learning agent consists of three basic elements, i.e., set of states (S) , action space (A) and reward function (R). At each decision step t the agent interacts with the current state $s_t \in S$, where S is a finite set of states, and takes an action $a_t \in A(s_t)$, where $A(s_t)$ is set available actions for the current state s_t and in return receives a reinforcement signal as feedback (reward) denoted by $r_t = R(s_t, a_t)$, which gives information about the quality of the action taken. The reward received by the agent is generally discounted over the given horizon by a discount factor denoted by γ (where, $0 < \gamma < 1$). This ensures that the reward received in the near future are given more importance as compared to reward received later.

A series of state-action pairs executed by the agent is known as policy, $\pi : S \rightarrow A$, which is a mapping from states to actions. For a given initial state (s_0) of the agent, the agent executes the first action $a_0 = \pi(s_0)$, and moves from state s_0 to s_1 and receives a reward $r_0 = R(s_0, a_0)$. This process continues until the end or until the goal is reached. The optimal solution to the MDP consists of a policy $\pi^* : S \rightarrow A$, which defines the set of desirable actions that needs to be executed at each state $s \in S$ in order to maximize the total cumulative

reward that the agent receives in the future. For a given arbitrary policy $\pi : S \rightarrow A$ from an arbitrary state s, the total discounted reward G_t could be written as shown in Eq. (16) below:

$$G_t = (R_t + \gamma . R_{t+1} + \gamma^2 . R_{t+2} + \ \ldots \mid s_t = s) \tag{16}$$

Here, γ represents the discount factor on the reward and determines the relative value of immediate reward vs future reward. The value of γ illustrates about how far the agent looks into the future, if the value of γ is set at 0 then the agents only tries to get the more immediate reward instead of looking into the future. Similarly, if the value of γ is set to 1, then the agent gives more importance to future reward as compared to the immediate ones. Now, in each state, there is a possibility of having more than one action that the agent can take. So in order to find the optimal policy, which is a particular combination of state-action pair, a function known as the state-action value function $Q(s, a)$ is defined. Here, $Q_\pi(s, a)$ shown in Eq. 17 indicates the utility of performing an action a at a given state s and following a policy π after that.

$$Q_\pi(s, a) = \left(\sum_{i=0}^{\infty \ or \ end} \gamma^i . R_{t+i} \middle| s_t = s, a_t = a \right) \tag{17}$$

Now, using the action-value function the optimal policy π^* can be expressed as:

$$\pi^*(s) = \arg\max_a \ Q(s, a) \tag{18}$$

One of the simple algorithms in solving model-free MDP problem is Monte Carlo Control Method (Sutton & Barto, 2018). Monte Carlo methods require the only experience, that is, sample sequence of state, corresponding actions and total rewards achieved. Here, each sequence or experience is termed as an episode. It is a way of solving RL problems based on averaging sample returns. As mentioned above, it is necessary to explicitly estimate the value or utility of each action at each state in order to suggest a policy. In order to determine the approximate true value of each state-action pair i.e. $Q(s, a)$, two alternating steps are performed namely policy evaluation and ε-greedy policy improvement (Sutton & Barto, 2018). The policy evaluation steps determine the Q-values (action-value function) for a given episode (sequence of state-action pairs) using Eq. (17); that is, it takes a policy π as input and returns the Q-values of each state-action pair in the policy π. Whereas, in ε-greedy policy improvement step, the policy is made greedy with respect to the current Q-values. Here ε-greedy policy means that most of the time a greedy action is selected during the policy

improvement step, but with probability ε a random action is selected instead of greedy selection, this is shown in Eq. (20). Note that the value of ε lies between 0 and 1. During the early stages of learning, the agent should select more random policies in order to encourage initial exploration i.e. large value of ε and, as time progresses, it should act more greedily i.e. very low value of ε. This is done in order to explore the state space during the starting of the method and exploit the state space during the end of the method (Thrun, 1992). The Monte Carlo learning method used to update the estimate of $Q(s,a)$ is shown in Eq. (21).

$$a(s) \leftarrow \begin{cases} \arg\max_a \ Q(s,a) & : \quad \text{with probability } 1-\varepsilon \\ \text{random action} & : \quad \text{with probability } \varepsilon \end{cases} \tag{20}$$

$$Q(s,a) := Q(s,a) + \alpha.\big(G_e - Q(s,a)\big) \tag{21}$$

Here in Eq. (21), $a(s)$ indicates the action that the agent needs to follow when it is at state s; $\alpha \in [0,1]$ represents the learning rate; and G_e represents the total cumulative reward that the agent receives at the end of each episode e.

6. The proposed algorithm

In this section, the Operational Aircraft Maintenance Routing Problem (OAMRP) is formulated as a Markov Decision using the knowledge provided in the previous section. Finally, the Monte Carlo based reinforcement learning algorithm is used in order to generate optimal routes that are maintenance fisible.

6.1 Formulation of OAMRP as Markov Decision Process

In this sub-section, the OAMRP is shown to be modelled as a sequential decision-making problem using the framework of Markov Decision Process (MDP). The three elements required for defining the MDP for OAMRP, i.e. state space, action space and reward function are illustrated in detail below.

6.1.1 State Space

In this paper a finite 4-dimensional state space is defined, where each state represents a particular flight leg. The flight leg state comprises the details of the particular flight leg's origin airport, arrival airport, departure time, arrival time. For example, a state $s = (A, B, 2{:}50, 4{:}00)$ indicates to a flight leg whose origin airport is A, destination airport is B, departure time is 2:50 and arrival time is 4:00. Hence, it should be noted that, the number of

available states is equal to the total number of available flight legs. The states are arranged in such a way that the departure time of each state i.e. flight leg, are in increasing order. The first state s_0 represents the first flight leg having the earliest departure time similarly the last state s_{end} represents the last flight leg. This sorting is done to define the sequence of the states in the MDP environment. The RL agent moves from the first state to the last state and assigns an aircraft tail number to each state of these states and in-turn collects reward if there is a through connect in between. The main objective is to maximize this reward collected by the agent. The details of the action space are explained below in the subtopic Action Space.

6.1.2 Action Space

In the proposed MDP model, the action of the agent at any decision step t is to assign a suitable aircraft tail number to the state s_t. Note that, here each state s represent a particular flight leg. The action space A for the RL agent at decision step t contains the tail numbers of the available aircrafts that can be assigned to the state $s_t \epsilon\, S$. The aircrafts available in the action space for a particular state or flight leg must be in accordance with the time and place constraints shown in Eq. (7) to Eq. (8) and maintenance constraints shown in Eq. (9) to Eq. (13). One example is provided below in order to understand the action space.

Let say the fleet size assigned for a particular cluster of flight legs by the airlines is K. The tail numbers for the aircrafts are indexed as 1, 2, 3. . . K. An example of action space for any arbitrary state s is $A(s) = \{2, 5, 6. . .k\}$ for $k \leq K$, which is a subset of the total number of aircraft that satisfies the time, place and maintenance constraints shown in section 4.2. That is, for each aircraft in the set $A(s)$, the destination airport of their previous state to which it was assigned earlier must be same as the origin airport of the current state s (place constraint), similarly the sum of the arrival time of each aircraft and Turn-Around-Time (TAT) must be earlier than the departure time of the current state s (time constraint) and each aircraft must also satisfy the maintenance constraints.

6.1.3 Reward Function

The reward function is used in accordance to the objective function that needs to optimized. In this paper, the main objective is to maximise the total through value or the total number of through connects. The reward function r_t is shown in Eq. (22) below, t is time step or decision step. The agent moves from one state to another state serially, and goes on assigning tail number to each of these states. If a through connect occurs after performing the action a

in state s_t with any of the previous states, then the agent receives a reward equivalent to the through value. Otherwise, the agent receives 0 reward.

$$r_t = \begin{cases} through\ value & : \text{if through connect occurs after taking action } a \text{ at } s_t \\ 0 & : \text{otherwise} \end{cases} \quad (22)$$

Here, r_t denotes the reward the agent gets after performing action a in the state s_t. The total cumulative reward received by the agent denotes the total through profit achieved by the airline and the main objective of this proposed method is to maximise this cumulative reward. Note that, in this paper the discount factor γ is taken to be 1 since the total cumulative rewards needs to be maximised.

6.2 Generating optimal routes using reinforcement learning

Using the above-formulated MDP framework and learning algorithm, the steps for the proposed Monte Carlo based reinforcement learning algorithm for OAMRP are explained in detail below:

Step 1: Prepare a list that contains the aircrafts' tail numbers (*K*). Prepare another list that includes the flight legs (FL) details, sorted with the increasing order of departure time of each flight. Number each flight leg in the *FL* list serially from 1 to *K*.

Step 2: Prepare a Q-value table with column as tail number and rows as flight leg number (flight leg number refers to the serial number, which is assigned in Step 1). Then initialise all the elements in the Q-table to value 0. Thus, the size of the Q-table becomes length(FL)*length(K). In the Q-table each element represents the value of taking action a in state s, that is, $Q(s, a)$. Initialize the number of episode e to 1.

Step 3: Split the list of aircraft, i.e. *K*, into three lists namely, an In Use (IU) list, Not In Use (NIU) list and Maintenance (MTN) list. Initially, the IU list and MTN list will be empty and the NIU list will be same as the list *K*, since no aircraft is in use. Once an aircraft is assigned to any state, then its corresponding tail number is removed from the NIU list and is added to the IU list. The IU list contains the details of the recent flight leg that the aircraft has travelled.

Step 4: Create two empty lists namely Normal Connect (*NC*) and Through Connect (*TC*)

Step 5: Start to create a complete policy, i.e. the sequence of state-action pairs. Each state here refers to a flight leg and the agent assigns an aircraft tail number to the current state and serially moves to the next state. In order to create a policy, start from state 1 and follow the following sub-steps until the last state or last flight leg:

Step a: If either list IU or list NC is empty then assign the first aircraft of the list NIU the current state, and move agent to the next state. Remove the aircraft from the NIU list, and add its detail to the IU list and move to step b. In addition, if any aircraft has completed maintenance, remove the tail number from MTN list and add it to NIU list, so that it can be used now.

Step b: If list IU is not empty then check whether there is any aircraft in the list IU, which satisfies the time and place, constraints with the agents current state i.e. Eq. (7) to Eq. (8) and the maintenance constraints in Eq. (9) to Eq. (10). If yes, then add the aircraft's tail number of the NC list and move to step c. Note that, there might me more than one aircraft which satisfies both the constraints, add all of these aircraft to the NC list. If no, return to step a. In addition, if there is any aircraft that does not follow the maintenance constraints remove the tail number of the aircraft from the IU list and add it to the MTN list. Then prepare a suitable maintenance check visit for the aircraft by taking into account the location constraints of the maintenance station as shown in Eq. (13) and workforce availability constraint shown in Eq. (11).

Step c: If list NC is not empty, then check whether any aircraft in the list NC satisfies the criteria for through connect. If yes, then remove those aircraft from NC add them to the TC list and move to step d. Note, there might be more than one aircraft that satisfy the through connect criteria; if so, add all of these aircraft to the TC list. If no, then move to step e.

Step d: With probability 1-ε, choose an aircraft from the TC having maximum Q-value for the current state among. That is if the current state is s then choose the aircraft k from TC having maximum $Q[s,k]$ in the Q-table and assign it to the current state. Or, with probability ε assign a random aircraft from the list TC to the current state. Then reward the agent according to the reward

function. Update the aircraft details in the IU list and move to the agent to the next state. Reset the lists NC & TC and move to step a.

Step e: With probability 1- ε choose the aircraft from the list NC and first aircraft of the NIU list (if NIU is not empty) having maximum Q-value for the current state among. That is if the current state is s then choose the aircraft k from list NC having maximum $Q[s,k]$ in the Q-table and assign it to the current state. Or with probability ε assign a random aircraft from the list NC or the first aircraft of the NIU list (if NIU list is not empty) to the current state. Then update the aircraft details in the IU list and move to the agent to the next state. Reset the list NC and move to step a.

Step f: If the agent completes its action in the last state after which no states are left, then move to Step 6

Step 6: Set the end of the episode. Calculate the total cumulative reward of the current episode e i.e. calculate the value of G_e, and update the Q-table according to the Eq. (21). Decrease the value of ε by a fixed factor η. Now increase the episodes by one and start from Step 3. In this paper, a suitable function is defined in order to find the value of G_e in Eq. (21) which is described below in Eq. (23). Note that the value of ε is decreased with increase in the number of episodes so that the agent takes more greedy approaches and hence results in convergence.

Step 7: Once the maximum allowed number of episodes (E_1) is reached, then terminate the process and act greedily with respect to the final Q-table obtained in order to find the optimal solution, that is, the optimum combination of flight legs and aircraft tail numbers.

The value of G_e used for updating the Q-value is shown in Eq. (23); it represents the total reward the agent gets at the end of each episode e.

$$G_e = \begin{cases} \sum_{t=1}^{t=S} r_t & : \quad 0 \le e \le E_2 \\ \left(\sum_{t=1}^{t=S} r_t\right) - G_{avg} & : \quad E_2 < e \le E_1 \end{cases} \tag{23}$$

Here, E_2 indicates the number of episodes after which we change the function of G_e. And, G_{avg} is the average of the total cumulative reward that the agent collected over E_2 episodes i.e.

$$G_{avg} = \left(\sum_{e=1}^{e=E_2} \sum_{t=1}^{t=S} r_t \right) \Big/ E_2 \tag{24}$$

Once the number of episodes reaches E_2, the Q-table is initialized again with all values equal to zero and is then updated using the new values of G_e. This is done in order to improve the rate of convergence of the algorithm by rewarding the agent if it gets more cumulative reward than the average reward up to episode E_2, i.e. G_{avg}, and penalizing the agent if it gets a less cumulative reward as compared to the G_{avg}.

It can be seen from the flowchart diagram (Fig. 3) that the proposed algorithm always follows a greedy way to earn maximum number of through connects wherever possible. If at a particular state only one through connect aircraft is available, then the agent assigns that aircraft to the state; and if more than one through connect aircrafts are available then the agent uses ε-greedy method to assign the aircraft. Moreover, if no through connect aircraft is available for a particular state then it assigns the aircraft in such a way that (according to the Q-values) it increases the possibility of more through connects in the near future. This is the basic formalism of reinforcement learning for measuring the utility of actions that gives no immediate reward but ensures better rewards in future.

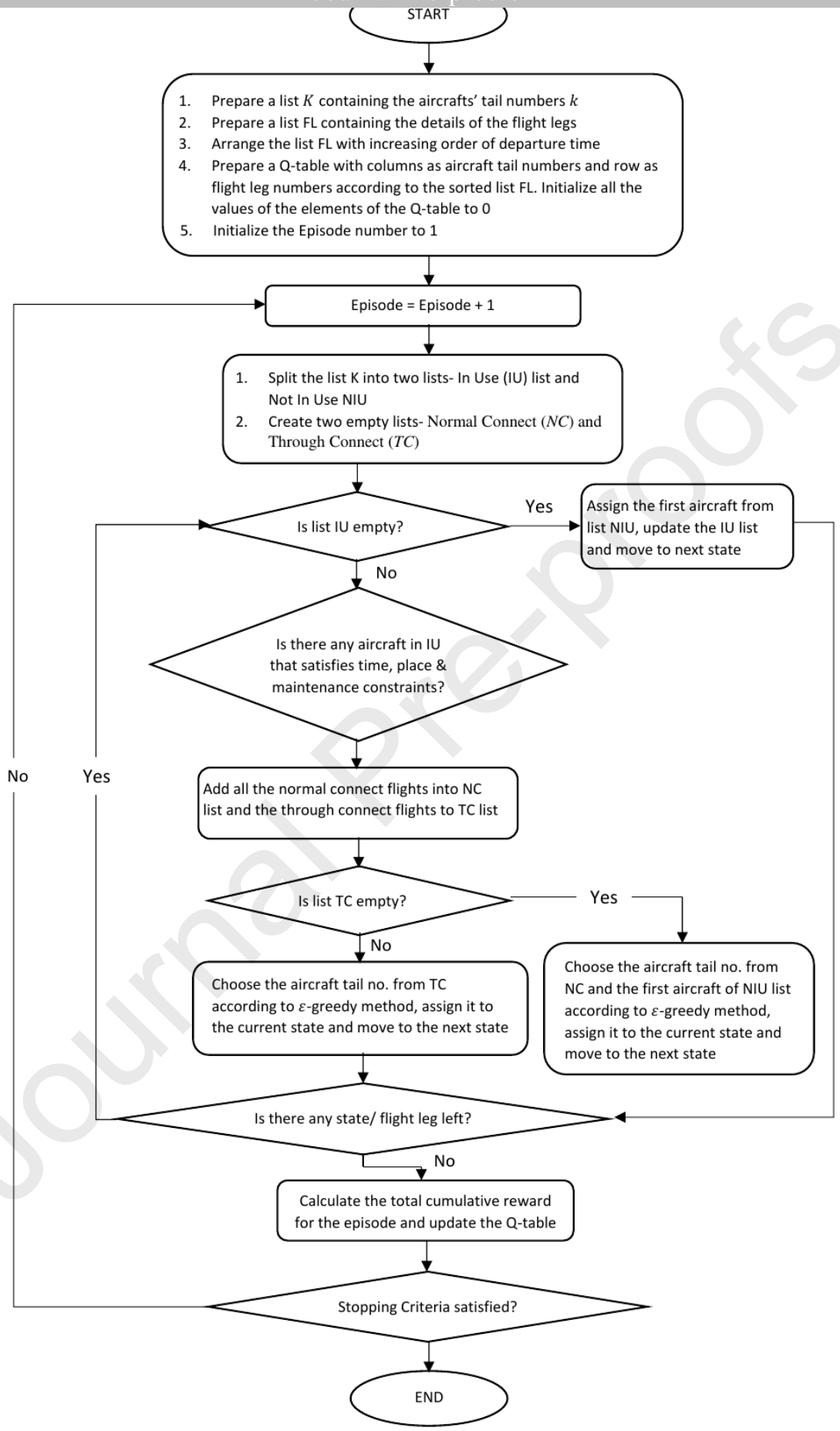

Fig. 3 Flowchart of the proposed

7. Experiments and results

The results generated for the OAMRP from both the reinforcement-learning based algorithm and from directly solving the ILP model is presented here. The ILP-based network flow model was solved by using CPLEX 12.1, a commercial software used by industries for optimization problems. The experiments were executed on the real case flight schedule data sets from the Egypt Air carrier. The size of the dataset varied from very small datasets (40 flight legs) to large case datasets (400 flight legs).

In case of the available commercial software like CPLEX, it is very challenging to obtain an optimum solution for medium and large size datasets as because the proposed OAMRP is combinatorial in nature and the computational time taken to obtain the optimal solution increases drastically for medium and large-sized dataset. Therefore, it is not possible to generate the exact solutions for all the cases. The main aim of the experiment is to verify the robustness and the effectiveness of the proposed learning method while solving real-world datasets.

Table 2. Details of the used dataset.

Cases	Flight-legs	Size of Fleet	Airports	Available maintenance stations	Take-off limit
1	40	8	4	4	10
2	48	7	5	4	7
3	64	8	7	4	7
4	96	14	13	6	10
5	160	11	10	6	15
6	240	26	19	9	15
7	296	30	26	9	15
8	400	42	28	9	15

7.1 Experimental setup

The real world flight schedule datasets used for experimentation was provided by EgyptAir. There are eight cases of flight schedules, each having different fleet size. The details about

each test case are shown in Table 2. As per EgyptAir, for all the test cases, the turn-around time is taken as 45 minutes and time required for maintenance is taken as 4 hours. As per the aircraft safety and FAA constraints we the maximum allowable flying-hour each aircraft to be in between 35 hours to 40 hours, i.e. $35 \text{ hrs} < T_{max} < 40 \text{ hrs}$. Also as per EgyptAir, through connect between two consecutive flight leg occurs if the transit time of the aircraft is between 45 minutes and 1.5 hours. The through connect revenue for all the cases is assumed to be 500. For the proposed algorithm, the value learning rate α was taken to be 0.1, if $G_{e+1} \leq G_e$ and 0.2 if $G_{e+1} > G_e$ and was kept the same throughout the experiment (i.e. the agent tries to learn more if the cumulative reward of the current episode is greater than the cumulative reward of the previous episode). The value of parameter ε was kept at 0.5 (i.e. 50% random search) during the first few iterations and was then decreased at a suitable rate η until the model converged. The minimum value of ε is equal to zero after which the RL agent will follow greedily according to the Q-values in order to obtain the optimum solution. Now for each test case the proposed algorithm was made to run 20 times and the average value of the obtained optimal solutions were reported. Note that, this iteration is not the same as that of the episode, for each iteration the algorithm has to run a particular number of episodes in order to obtain the converged solutions. The proposed algorithm was coded in Python 3.6 platform and the ILP was coded in CPLEX 12.1 and the experiment was carried out on desktop PC with 2.40 GHz, Intel(R) Core(TM) i7-5500U CPU, 8.0 GB RAM and Microsoft Windows 10 operating system (64-bit).

7.2 Performance of both algorithms while solving small sized test cases

Most of the airlines generally consider 200-250 flight leg schedules to be small-sized instance. In the data provided by the EgyptAir (Table 2), test cases up to 240 flight legs schedule and 26 aircraft are considered to be small-sized test cases, i.e. first six cases of Table 2 are considered to be small-sized test cases. The results obtained by the CPLEX 12.1 and the proposed algorithm for small sized test cases are shown in Table 3. The results obtained from solving the mathematical model in CPLEX 12.1 is shown in the first two columns of Table 3, here OS represents the optimal solution obtained and $Time$ determines the time taken to obtain the solution, i.e., computational time. Furthermore, the rest of the columns of Table 3 summarizes the results obtained from the proposed algorithm, where S_{Best} column represents the best optimal solution obtained by the proposed algorithm among the 20 iterations, $\bar{S}$ represents the average of all the solutions obtained by the proposed

algorithm and σ denotes the standard deviation among the obtained solutions. Note that, here, $\overline{Time}$ is the average time taken by the proposed method for each iteration and not the time taken to obtain the best optimal solution. In order to compare the quality of solution obtained by CPLEX 12.8 and the proposed method, the parameter $\%GAP$ is calculated in table 3 as the performance indicator, where $\%GAP = (OS - \bar{S})/ OS$.

Table 3. Performance of both the algorithms while solving small-sized test cases.

Cases	CPLEX		Proposed Algorithm				$\%GAP$
	OS	$Time$ (s)	S_{Best}	$\bar{S}$	σ	$\overline{Time}$ (s)	
1	3500	4.12	3500	3500	0	0.08	0
2	0	6.56	0	0	0	0.09	0
3	6000	27.42	6000	6000	0	0.21	0
4	4000	64.59	4000	4000	0	0.67	0
5	18000	365.21	18000	17975	108.973	1.19	0.138
6	25500	8561.65	25500	25425	178.536	1.75	0.294

It can be seen from Table 3 that, the values of both $\bar{S}$ and S_{Best} obtained by running the proposed algorithm remains equal to the values of OS obtained from CPLEX for the first four test cases. With the increase in the number of flight legs and number of aircrafts (last two cases of Table 3), the value $\bar{S}$ deviates from the value of OS, but S_{Best} stills remains equal to the OS. From the value of standard deviation, it can be interpreted that, there is no difference in the value of the optimal solution obtained by the proposed algorithm and CPLEX for first four test cases. However, the difference increases slightly for the last two test cases by at most 0.29%. Thus, the stability and the accuracy of the proposed method is verified in comparison to CPLEX for small scale datasets.

Regarding computational time, it can be interpreted from the Table 3 that the average time taken by the proposed reinforcement learning algorithm is that it takes much less time as compared to CPLEX (i.e. at most four seconds). But the time taken by the CPLEX increases very drastically as the number of flight legs and the number of aircraft are increased, whereas there is no drastic change in the time taken by the proposed method for obtaining the optimal solution. For example, in case 6, CPLEX takes around two hours to get the optimum solution whereas the proposed algorithm takes only about four seconds. Fig. 4 compares the time

taken by CPLEX and the proposed reinforcement-learning algorithm for obtaining the optimum solution in case of small-sized test cases.

Hence, from the above discussion, it can be concluded that the proposed reinforcement learning based algorithm obtains the optimum solution much faster than the CPLEX for small sized test cases. In terms of the obtained solution quality, it can be concluded that the proposed algorithm generates high quality solution since the best solution (S_{Best}) always remains equal to the optimum solution and the average solution (average over 20 iterations) deviates by at most 0.29% from the optimal solution.

In order to determine the robustness and efficiency of the proposed algorithm, small test instances up to 240 flight legs and 26 aircrafts are not sufficient. However, small-scale datasets were useful in comparing the performance of the proposed method with the exact methods. In case of large and medium scale datasets, however, CPLEX failed to generate even a viable solution in a reasonable amount of time and memory. The comparison between CPLEX and the proposed method in case of large and medium-sized test cases is shown in the next sub-section.

7.3 Performance of both the algorithms while solving large and medium-sized test cases

In this subsection, the proposed method is made to run on large-scale datasets in order to assess the robustness and the applicability of the algorithm to a real world problem. The test cases 7 and 8 from Table 2 are considered to be larger as compared to others. Since for larger cases CPLEX failed to obtain an optimum solution in a reasonable amount of memory and time, it is proposed to obtain the best Upper-Bound solution from CPLEX in order to carry on the comparison. To obtain the best Upper-Bound the maximum running time for CPLEX was set to 7 hours, since longer running times of CPLEX ensures a better bound to the solution. The summary of the solutions obtained from both models is shown in Table 4 using the same notations as that of Table 3.

Table 4. Performance of both the algorithms while solving large and medium-sized test.

Cases	CPLEX Upper-Bound	Proposed Algorithm				$\%GAP$
		S_{Best}	$\bar{S}$	σ	$\overline{Time}$ (s)	
7	32000	32000	31825	238.48	2.49	0.550
8	63000	63000	62575	435.89	6.62	0.675

It can be seen from the Table 4 that the proposed method still generates very good solutions for large-scale datasets in a very small amount of time. Moreover, the S_{Best} still remains the same as that of the obtained Upper-Bound from CPLEX, however the average solution $\bar{S}$ deviates from the Upper-Bound by at most 0.675%. It is worth noting that even for very large dataset (case 8) the order of time complexity still remains as seconds where as it is in order of hours for the CPLEX. Hence, this shows that the proposed algorithm is also very fast and efficient while solving large real world test cases. From the standard deviation, it can be inferred that that proposed algorithm has very low solution variability, which confirms that the algorithm is stable and reliable.

7.4 Comparison of the proposed algorithm with other available methods in the literature

The performance of the proposed algorithm was compared with CPLEX while solving the aircraft maintenance routing problem in the previous few sub-sections. Simply comparing the efficiency of the proposed algorithm with CPLEX is insufficient to justify its superiority over the other available methods in the literature. There are many algorithms in the literature that tend to solve the aircraft maintenance routing problem in polynomial time. Hence, in this sub-section we compare the order of computational time and quality of the solution obtained by the proposed reinforcement learning algorithm with three other commonly available methods in the literature, namely- Genetic Algorithms (GA), Ant Colony Optimization (ACO) and Simulated Annealing (SA). For this purpose, the approximate values of the computational time and solution quality for GA, ACO and SA based algorithms are taken from the experiments conducted by Eltoukhy et al. (2017). The experimental result of the computation times are summarized in Table 5, and the solution qualities are summarized in Table 6.

It can be interpreted from the Table 5 that the method based on SA is much better as compared to ACO and GA in producing an optimal solution, since it produces solution even for large-scale dataset in approximately 80 seconds. However, among the three GA has a high order of computational time. It takes at most 24 minutes for generating solution for the large-scale dataset, whereas ACO takes 150 seconds and SA takes 80 seconds. Even for the very small-scale dataset (48 flight legs and 8 aircrafts), it takes around 12 seconds whereas GA and SA take around 1 second approximately and CPLEX itself takes around 4 seconds (Table 3). However, it can be clearly seen from Table 6 that, proposed reinforcement learning based

algorithm outperforms the other three methods. It generates its solution within, at most, 7 seconds even for very large-scale datasets (400 flight legs and 42 aircrafts). The asymptotic order of time taken by the proposed model for all the test cases are very small as compared to others which verifies that the proposed algorithm is computationally more efficient than available methods in literature other than CPLEX. Fig. 4 summarizes the overall comparison between the computational times of different available methods.

Table 5. Comparison between computational times of other available algorithms.

Cases	Dataset Size	Computational Time (s)			
		Ant Colony Optimization (ACO)	Simulated Annealing (SA)	Genetic Algorithm (GA)	Proposed RL Algorithm
1	40	1.30	0.87	12.28	0.08
2	48	1.73	1.23	15.05	0.09
3	64	4.85	2.07	31.40	0.21
4	96	6.50	3.62	44.40	0.67
5	160	24.29	11.82	81.99	1.19
6	240	38.07	33.91	326.45	1.75
7	296	63.87	43.59	368.18	2.49
8	400	147.98	81.95	1428.51	6.62

Table 6. Comparison between solution qualities obtained from other available methods.

Cases	Dataset Size	Solution quality (%GAP)			
		Ant Colony Optimization (ACO)	Simulated Annealing (SA)	Genetic Algorithm (GA)	Proposed RL Algorithm
1	40	0	0	0	0
2	48	0	0	0	0
3	64	4.85	4.77	4.29	0
4	96	3.04	0.17	0	0
5	160	11.91	5.69	5.46	0.138
6	240	12.30	7.57	6.40	0.294
7	296	9.47	4.68	4.43	0.550
8	400	16.43	8.67	8.09	0.675

As described in the previous subsections that the $\%GAP$ is used as the performance indicator of a model which states how much the average solution of the model deviates from the optimal solution obtained from CPLEX. From the solution quality table 6 it can be seen that among ACO, SA and GA, GA generates better solution quality as compared to the other two. In case of the very small-scale dataset (case 1 and case 2) the three algorithms produce the exact optimal solution. However, with an increase in the size of the dataset, the quality of the solution obtained by the three methods decreases very rapidly. For very large-scale dataset, the method based on ACO generates solution with very large deviation from the exact solution i.e. up to 16% whereas GA and SA produces solution with deviation up to 8%. However, as interpreted from the tables that the proposed algorithm produces high quality solution even for very large-scale dataset. For small-scale datasets from case1 to case 4 the proposed algorithm produces the exact result with 0% deviation and for very large-scale datasets, the proposed algorithm produces solution with deviation only up to 0.7% from the exact solution which is very less as compared to the other three algorithms. Hence, this overall verifies that the proposed algorithm in not only computationally efficient as compared to other algorithms available in the literature but also produces very high quality solution.

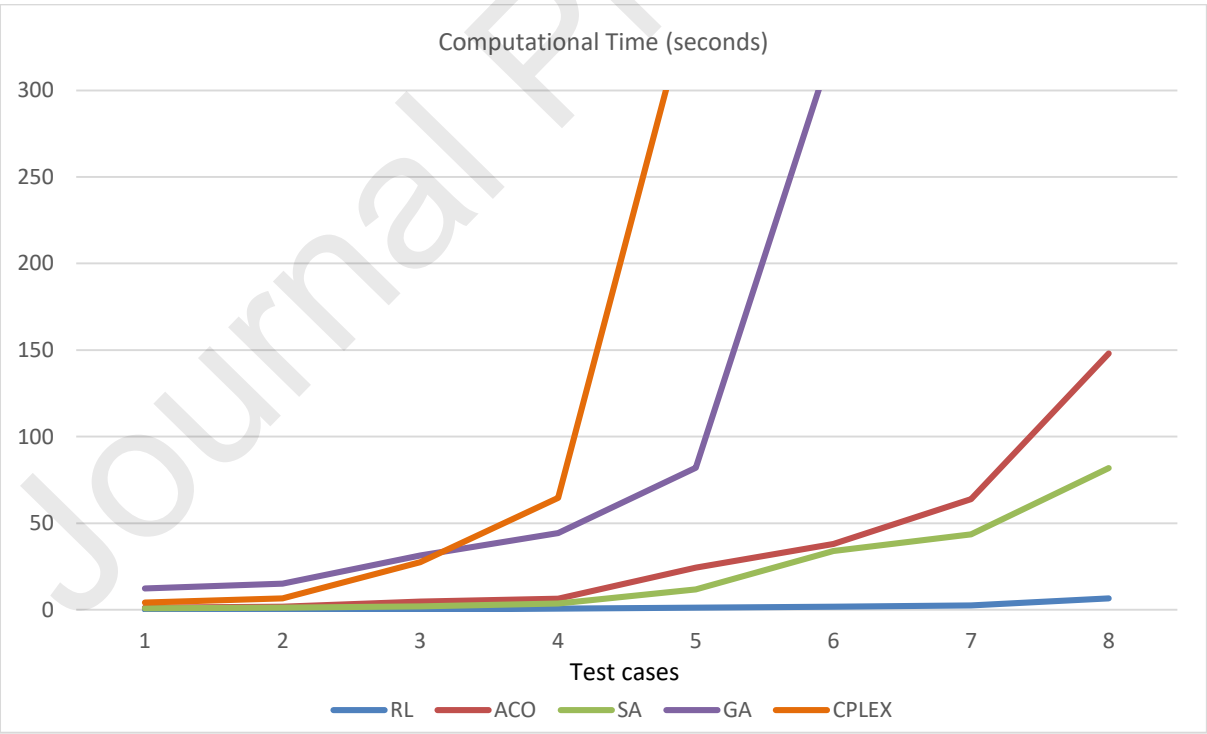

Fig. 4 Comparison of computational times among different available methods.

7.5 Convergence of proposed model to optimal solution

The first thought that arises while applying a new model to solve a given optimization problem is that, whether the model is converge to an optimal solution or not. Monte Carlo method of reinforcement learning is considered as the basic method of reinforcement learning. As described in section 5, the ε-greedy policy improvement step of the reinforcement learning helps in generating optimal sequence of state-action pairs. In this step, the agent takes either a greedy policy based on the present Q-values of each state with probability 1- ε or takes a random policy with probability ε. This ensures exploiting the presently known Q-values in order to maximise reward and at the same time it ensures exploration in order to search new better policy. This in return helps the agent in order to find the optimal sequence of state-action pair .The reference to the mathematical prove of the convergence of Monte Carlo method of reinforcement can be found in the Chapter 5 of Reinforcement learning: An introduction (Sutton & Barto 2018). This subsection presents how the convergence of the solution to the optimal policy is affected by the parameter ε of ε-greedy policy improvement step and the total number of iterations or episodes.

The effect of the parameter ε on the convergence of solution towards the optimum solution is shown in Fig. 5. Three cases of table 2 were used, i.e., case 5, case 7 and case 8. Initially the value of epsilon (ε) is taken to be 0.5, where the agent takes 50% greedy and 50% random steps in order to explore the solution space. With the increase in number of iterations, the value of epsilon is decreased at a suitable rate (η) resulting in more number of greedy steps and finally the result converges to the optimal solution, as the value of epsilon nears the value zero. If the rate (η) kept too low, then the value of ε may remain fixed or decreases at a very slower rate as a result of which the agent will keep on taking random steps throughout the episodes and won't be able to converge to the optimum solution. This is illustrated for case 5 and case 8 of table 2 in figure 6. It can be seen from the figure 6, that the agent keeps on exploring the solution space without exploiting and generating the optimal solution if parameter ε is kept fixed at 0.5 throughout all the episodes. Similarly, if the rate (η) is kept too high, the value of ε ill decrease sharply to the value zero. As a result, the agent will keep on exploiting few solutions without exploring the solution space, which will lead to pre-mature convergence. This will result in sub-optimal solution. This is depicted in figure 7. In addition, the impact of total number of iterations on the total cumulative reward is shown in

figure 8. The figure is obtained by running the model for different number of iteration and then checking the change in total cumulative reward. It can be clearly seen from figure 8 that, sufficient number of iterations are required by the agent in order to arrive at the optimal solution by exploring and exploiting the solution space. Less number of iterations might lead to pre-mature convergence or non-convergence of the model.

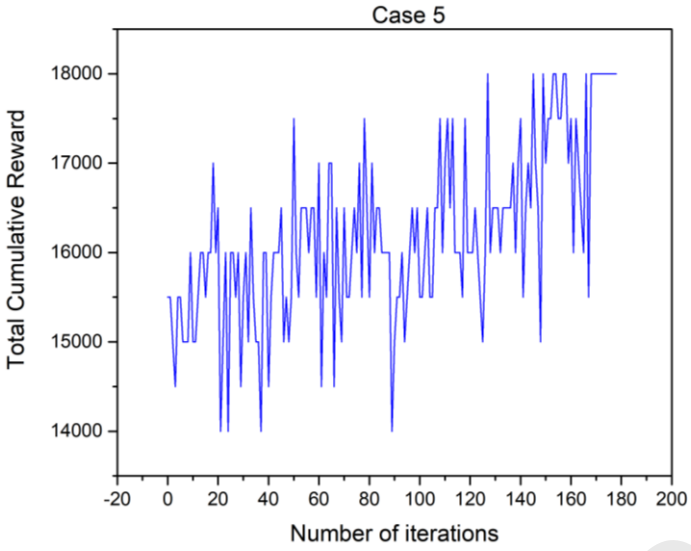

Fig. 5a

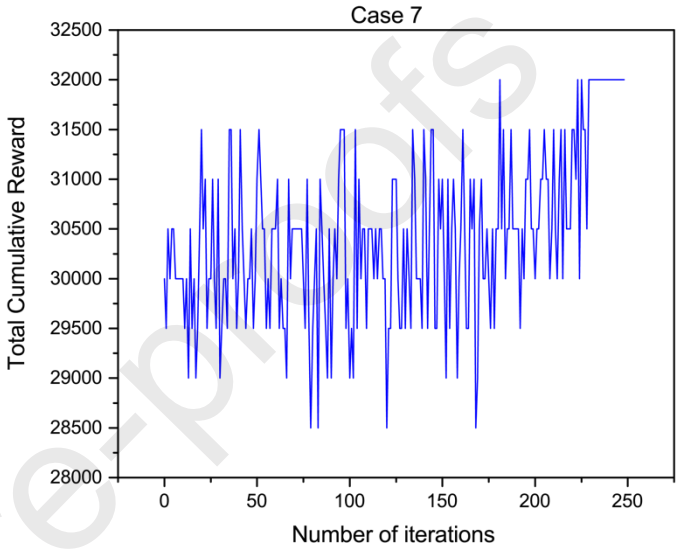

Fig. 5b

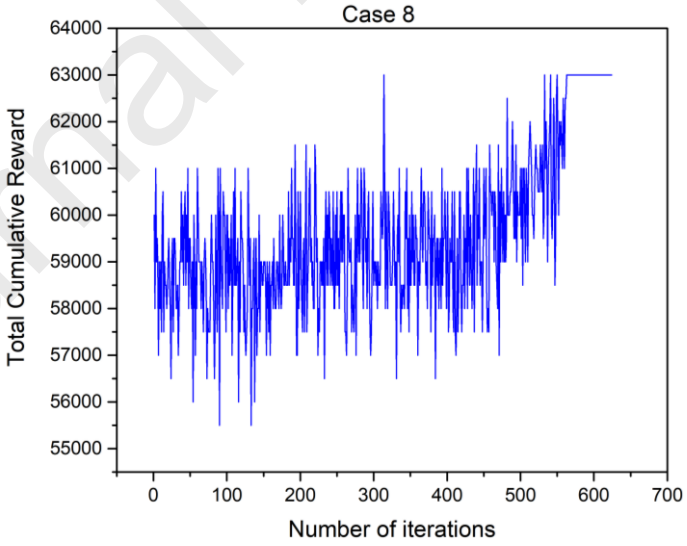

Fig. 5c

Fig. 5. This illustrates the reinforcement-learning model converges to an optimal solution if the value of the parameter ε is decreased at a suitable rate of η. In the first half, the agent takes random steps, in the second half as the value of ε nears zeros the agent takes steps greedily which results in convergence of the model.

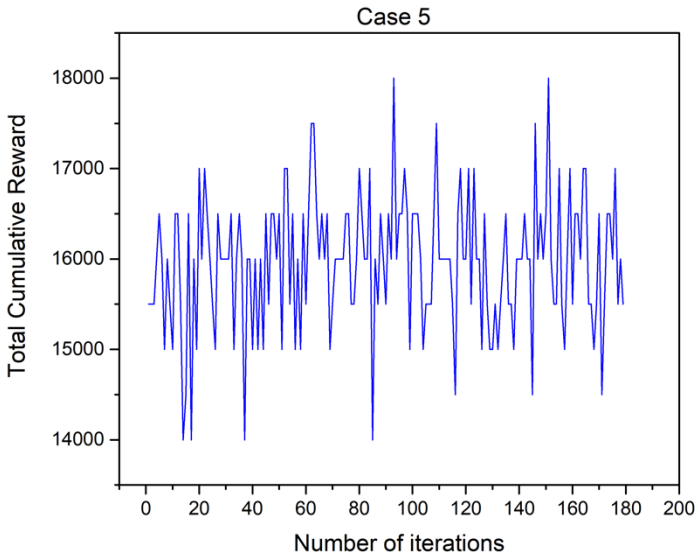

Fig. 6a

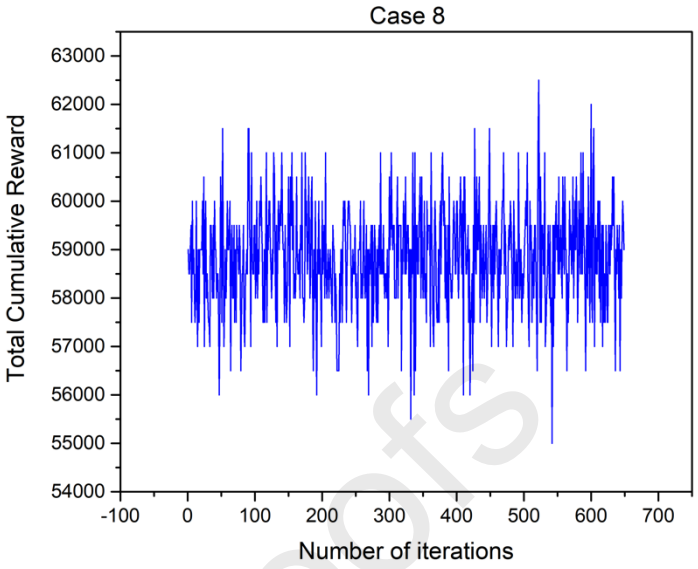

Fig. 6b

Fig. 6. This illustrates that the model do not converge to an optimal solution if the value of the parameter ε is either kept fixed at 0.5 or is reduced at a very slower rate. The agent will keep on exploring the solution space without conversing to an optimal solution.

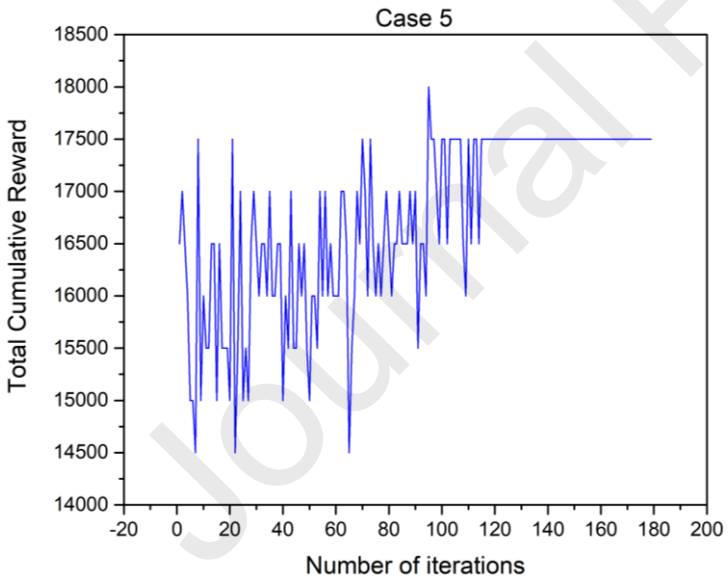

Fig. 7a

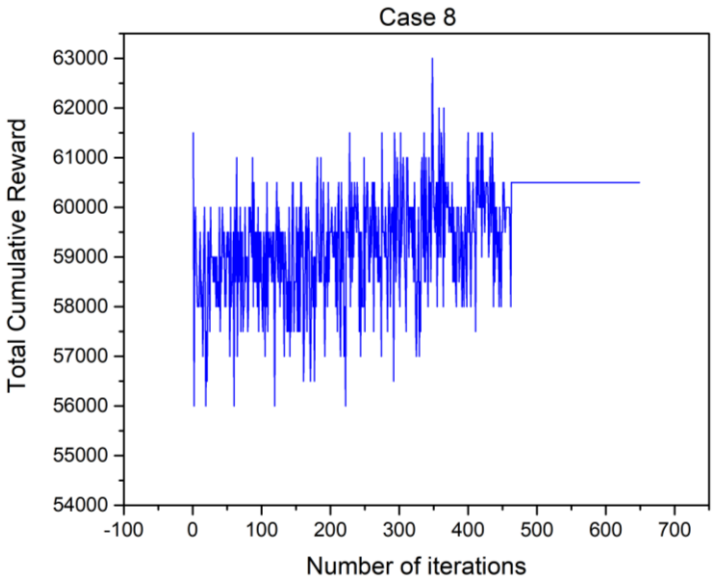

Fig. 7b

Fig. 7. This illustrates that the model do not converge to an optimal solution if the value of the parameter ε is decreased at a faster rate. The agent will exploit the initial solutions without exploring the solution

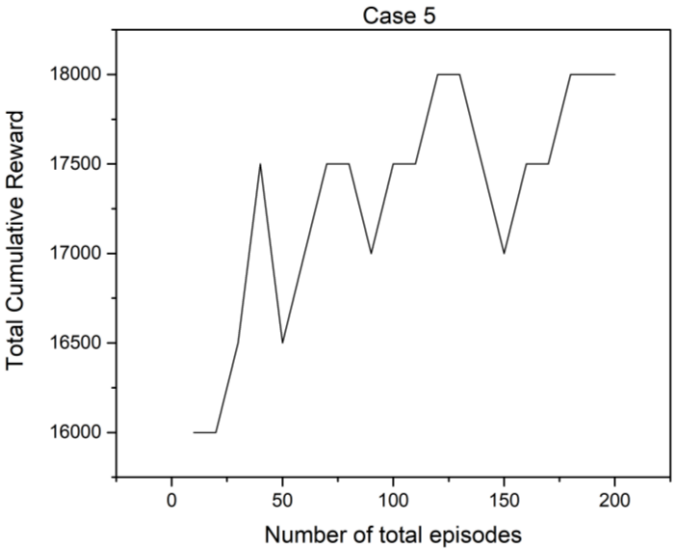

Fig. 8a

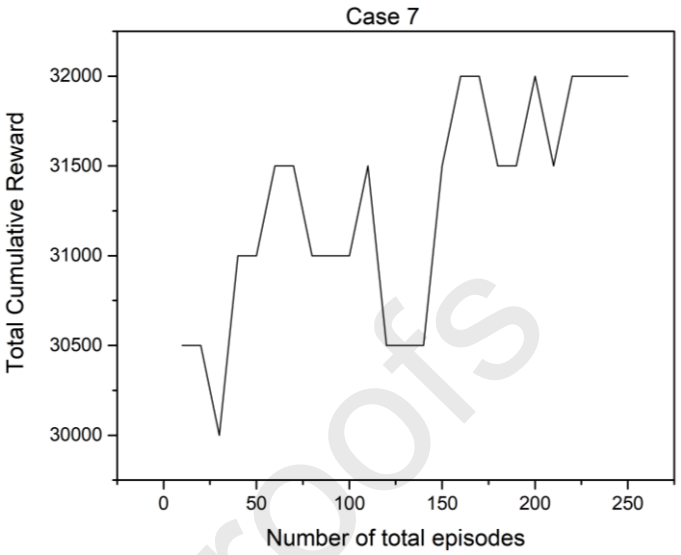

Fig. 8b

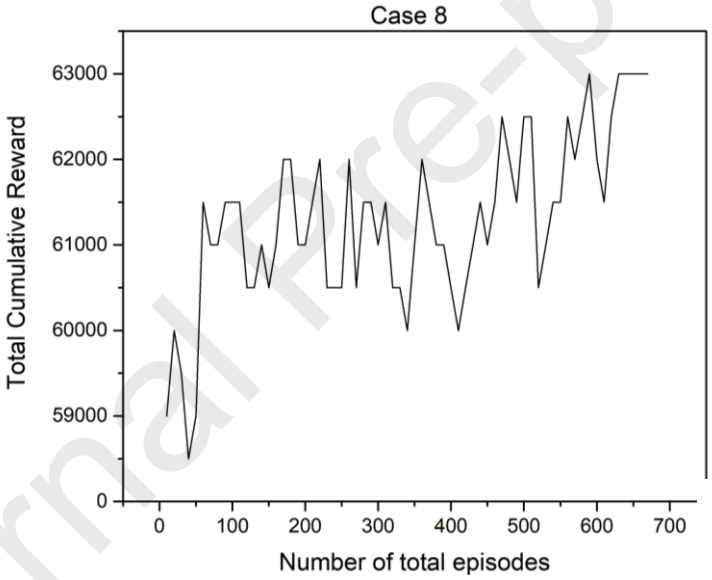

Fig. 8b

Fig. 8c

Fig. 8. This illustrates the impact of total number of episodes on total cumulative reward i.e. the value of total cumulative reward obtained when the model is ran over different number of total episodes. It is ensured that in each run the value of ε is decreased from 0.5 to 0 at a suitable rate

The main reasons for the outperformance of the proposed model in terms of quality of the solution obtained and speed is summarized in detail below.

1. *Quality of the solution obtained*: The proposed algorithm follows a greedy way to solve the OAMRP, as a result, at every step the algorithm always tries to maximize the number of through connects. The agent in the algorithm always generates policies

by taking actions with high Q-values, which in turn helps it in assigning aircrafts to states that might not be having an immediate reward (through connect), but would increase the chances of future rewards. This kind of memorization property is absent in other available methods which helps in getting high quality solutions. As mathematically proved by Sutton and Barto (2018), and from the above obtained experimental results, the reinforcement learning algorithm always converges to the optimum solution after performing a number of sample episodes or iterations.

2. *Computational time*: Unlike many heuristic-based solution methods presented in the literature that iterate their algorithm randomly on the feasible solution space, the proposed algorithm uses a learning paradigm in order to improve the time required for convergence. The agent in the reinforcement learning algorithm always tries to learn from each of its previous iterations or episodes, and in return updates the Q-values of each state-action pair in the Q-table, that is later used in the future episodes to generate optimal policy. In addition, many meta-heuristic based methods such as ACO, SA etc. use complex calculations in order to find the fitness values or quality of solution generated in each iteration at the expense of computational time. Whereas, in the proposed algorithm the fitness value or quality of solution generated in each iteration is the total through connect value, which is generated by a simple sum of each individual through value.

3. *Model complexity*: The proposed algorithm is much simpler and concise as compared to other algorithms in the literature. It simply uses a Q-table that is updated using the experience generated in each iteration, and uses it to determine greedy policy for future iterations. Additionally, the number of parameters used in the model are much fewer as compared to other models and can be tuned very easily and efficiently, whereas tuning model parameters of other models is quiet computationally expensive. The solution generated also depends largely on the values of model parameters and having a small number of parameters gives an advantage of getting high quality of solution.

8. Conclusion and future work

In this paper, a new ILP formulation for the 4-day operational aircraft maintenance routing problem was presented along with an effective reinforcement learning based algorithm for getting high-quality solutions in very less computational time. No approximations were assumed while dealing with the maintenance constraints and without rejecting the work-force availability constraint.

In terms of solution method, the ILP formulation was first solved using the commonly available software named CPLEX that produced exact solutions for small sized datasets. In case of large-scale dataset the software failed to generate the exact solution, and hence the best Upper-Bound solution was generated after running it for 7 hours. Also in the case of small dataset CPLEX took a considerable amount of time and memory. Second, in order to avoid long computational times and heavy usage of memory, an effective and efficient algorithm was proposed based on reinforcement learning that produced high-quality solution in very less time. The performance and the efficiency of the proposed algorithm was compared to that of CPLEX and other heuristic algorithms like Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Simulated Annealing (SA) (Eltoukhy et al., 2017). It was found out that the best solutions generated by the proposed algorithm were the same as that of the exact solutions obtained from CPLEX in case of small-scale dataset whereas the average solution deviated slightly (0.29%) from the exact solution. For large-scale dataset, the solution obtained from the proposed algorithm was equal to the Upper-Bound solution obtained from CPLEX and the average solution deviated by at most 0.675% from the Upper-Bound solution. Whereas, the solutions of ACO, SA and GA algorithms deviated by 16.43%, 8.67% and 8.09% respectively as compared to CPLEX in case of large-scale dataset. Hence, the proposed algorithm obtained high quality solutions that were very near to the exact solutions obtained by CPLEX. With respect to the computational time, the proposed algorithm takes very less amount of time as compared to other algorithms. It takes less than 2 seconds for finding the best solution in case of small and medium scale dataset whereas CPLEX takes around 2.3 hours. On the other hand, for large-scale datasets CPLEX fails to produce the best solution whereas the proposed algorithm generates the best solution equal to the Upper-Bound within 7 seconds and the average solution deviates by 0.675%. Hence, with respect to the computational time, the proposed algorithm is shown to have a very fast performance as compared to other available methods.

One of the drawbacks of the proposed OAMRP and the solution algorithm is that it did not take any considerations about the unfavourable events such as disruptions in flight schedule.

Since disruptions are very common in the airline industry, one such direction for future research is to create an efficient model that provides good solutions during such events.

Acknowledgment

The work described in this paper was supported by a grant from the National Natural Science Foundation of China (Grant No. 71901052).

References

Al-Thani, N. A., Ahmed, M. B., & Haouari, M. (2016). A model and optimization-based heuristic for the operational aircraft maintenance routing problem. *Transportation Research Part C: Emerging Technologies*, *72*, 29-44.

Aydin, M.E. and Öztemel, E., 2000. Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, *33*(2-3), pp.169-178.

Ball, M., Barnhart, C., Nemhauser, G., & Odoni, A. (2007). Air transportation: Irregular operations and control. *Handbooks in operations research and management science*, *14*, 1-67.

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. *Transportation Science*, *32*(3), 208–220.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W. and Vance, P.H., 1998. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, *46*(3), pp.316-329.

Başdere, M., & Bilge, Ü. (2014). Operational aircraft maintenance routing problem with remaining time consideration. *European Journal of Operational Research*, *235*(1), 315–328.

Bharti, S., Kurian, D.S. and Pillai, V.M., 2020. Reinforcement Learning for Inventory Management. In *Innovative Product Design and Intelligent Manufacturing Systems* (pp. 877-885). Springer, Singapore.

Chen, L., Gendreau, M., Hà, M. H., & Langevin, A. (2016). A robust optimization approach for the road network daily maintenance routing problem with uncertain service

time. *Transportation research part E: logistics and transportation review*, *85*, 40-51.

Chen, J., Yuan, B. and Tomizuka, M., 2019, October. Model-free deep reinforcement learning for urban autonomous driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)* (pp. 2765-2771). IEEE.

Clarke, L., Johnson, E., Nemhauser, G., & Zhu, Z. (1997). The aircraft rotation problem. *Annals of Operations Research*, *69*, 33-46.

Eltoukhy, A. E., Chan, F. T., & Chung, S. H. (2017). Airline schedule planning: a review and future directions. *Industrial Management & Data Systems*, *117*(6), 1201-1243.

Eltoukhy, A. E., Chan, F. T., Chung, S. H., & Niu, B. (2018). A model with a solution algorithm for the operational aircraft maintenance routing problem. *Computers & Industrial Engineering*, *120*, 346-359.

Eltoukhy, A. E., Chan, F. T., Chung, S. H., Niu, B., & Wang, X. P. (2017). Heuristic approaches for operational aircraft maintenance routing problem with maximum flying hours and man-power availability considerations. *Industrial Management & Data Systems*, *117*(10), 2142-2170.

Etschmaier, M. M., & Mathaisel, D. F. (1985). Airline scheduling: An overview. *Transportation Science*, *19*(2), 127-138.

Gabel, T. and Riedmiller, M., 2012. Distributed policy search reinforcement learning for job-shop scheduling tasks. *International Journal of production research*, *50*(1), pp.41-61.

Gopalakrishnan, B., & Johnson, E. L. (2005). Airline crew scheduling: state-of-the-art. *Annals of Operations Research*, *140*(1), 305-337.

Gopalan, R., & Talluri, K. T. (1998). The aircraft maintenance routing problem. *Operations Research*, *46*(2), 260-271.

Gosavi, A. (2009). Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, *21*(2), 178-192.

Govindan, K., Jafarian, A., Khodaverdi, R., & Devika, K. (2014). Two-echelon multiple-vehicle location–routing problem with time windows for optimization of sustainable supply chain network of perishable food. *International Journal of Production*

Economics, *152*, 9-28.

Haouari, M., Shao, S., & Sherali, H. D. (2012). A Lifted Compact Formulation for the Daily Aircraft Maintenance Routing Problem. *Transportation Science*, *47*(4), 508–525.

Kabbani, N. M. & Patty, B. W. (1992). Aircraft routing at American airlines. In Proceedings of the 32nd annual symposium of AGIFORS. Budapest, Hungary.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kara, A. and Dogan, I., 2018. Reinforcement learning approaches for specifying ordering policies of perishable inventory systems. *Expert Systems with Applications*, *91*, pp.150-158.

Khamis, M. A., & Gomaa, W. (2014). Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. *Engineering Applications of Artificial Intelligence*, *29*, 134–151.

Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A.A.A., Yogamani, S. and Pérez, P., 2020. Deep reinforcement learning for autonomous driving: A survey. *arXiv preprint arXiv:2002.00444*.

Lee, L. H., Lee, C. U., & Tan, Y. P. (2007). A multi-objective genetic algorithm for robust flight scheduling using simulation. *European Journal of Operational Research*, *177*(3), 1948–1968.

Liang, Z., Chaovalitwongse, W. A., Huang, H. C., & Johnson, E. L. (2011). On a New Rotation Tour Network Model for Aircraft Maintenance Routing Problem. *Transportation Science*, *45*(1), 109–120.

Mak, V., & Boland, N. (2000). Heuristic Approaches to the Asymmetric Travelling Salesman Poblem with Replenishment Arcs, *7*, 431–447.

Orhan, I., Kapanoglu, M., & Karakoc, t. H. (2011). Concurrent Aircraft and Maintenance Scheduling. *Journal of Aeronautics and Space Tecnologies*, *5*(1), 73–79.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rennie, J., & McCallum, A. K. (1999). Using reinforcement learning to spider the Web efficiently. *Proceedings of ICML-99, 16th International Conference on Machine Learning*, 335–343.

Safaei, N. and Jardine, A.K., 2018. Aircraft routing with generalized maintenance constraints. *Omega*, *80*, pp.111-122.

Sarac, A., Batta, R., & Rump, C. M. (2006). A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research*, *175*(3), 1850–1869.

Šemrov, D., Marsetič, R., Žura, M., Todorovski, L., & Srdic, A. (2016). Reinforcement learning approach for train rescheduling on a single-track railway. *Transportation Research Part B: Methodological*, *86*, 250–267.

Sherali, H. D., Bish, E. K., & Zhu, X. (2006). Airline fleet assignment concepts, models, and algorithms. *European Journal of Operational Research*, *172*(1), 1–30.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser et al. "Mastering the game of Go with deep neural networks and tree search." *nature* 529, no. 7587 (2016): 484-489.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert et al. "Mastering the game of go without human knowledge." *nature* 550, no. 7676 (2017): 354-359.

Sriram, C., & Haghani, A. (2003). An optimization model for aircraft maintenance scheduling and re-assignment. *Transportation Research Part A: Policy and Practice*, *37*(1), 29–48

Sutton, R.S. and Barto, A.G., 2018. *Reinforcement learning: An introduction*. MIT press.

Thrun, S. B. (1992). Efficient exploration in reinforcement learning.

Walraven, E., Spaan, M. T. J., & Bakker, B. (2016). Traffic flow optimization: A reinforcement learning approach. *Engineering Applications of Artificial Intelligence*, *52*, 203–212.

Wang, Y.C. and Usher, J.M., 2005. Application of reinforcement learning for agent-based

production scheduling. *Engineering Applications of Artificial Intelligence*, *18*(1), pp.73-82.

Yan, S., Tang, C. H., & Lee, M. C. (2007). A flight scheduling model for Taiwan airlines under market competitions. *Omega*, *35*(1), 61–74.

Zhang, Z., Wang, W., Zhong, S. and Hu, K., 2013. Flow shop scheduling with reinforcement learning. *Asia-Pacific Journal of Operational Research*, *30*(05), p.1350014.

Zhu, F., & Ukkusuri, S. V. (2015). A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment. *Transportation Research Part C: Emerging Technologies*, *55*(2015), 363–378.

Zolfpour-Arokhlo, M., Selamat, A., Mohd Hashim, S. Z., & Afkhami, H. (2014). Modeling of route planning system based on Q value-based dynamic programming with multi-agent reinforcement learning algorithms. *Engineering Applications of Artificial Intelligence*, *29*, 163–177.

Research Highlight:

- Development of ILP model for the operational aircraft maintenance routing problem.
- Development of a new reinforcement learning based algorithm to solve the problem.
- Flying hrs; no. of take-off; workforce capacity are major maintenance constraints.

Author contributions

J.H. Ruan: Conceptualization, Methodology, Software, Investigation, Writing Original Draft.

Z.X. Wang: Validation, Formal analysis, Visualization, Software.

Felix T.S. Chan: Validation, Formal analysis, Visualization.

S. Patnaik: Resources, Writing, Review & Editing, Supervision, Data Curation.

M.K. Tiwarif: Resources, Writing, Review & Editing, Supervision, Data Curation.