https://doi.org/10.1016/j.cie.2018.08.012

Joint optimization using a leader-follower Stackelberg game for coordinated configuration of stochastic operational aircraft maintenance routing and maintenance staffing

Abdelrahman E.E. Eltoukhy[a], Z.X. Wang[b,*], Felix T.S. Chan[a], S.H. Chung[a]
[a]*Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hum, Hong Kong*
[b] *School of Business Administration; Institute of Supply Chain Analytics, Dongbei University of Finance and Economics, Dalian, China*

*Corresponding Author: Z.X. Wang

elsayed.abdelrahman@connect.polyu.hk
wangzhengxu@dufe.edu.cn
f.chan@polyu.edu.hk
mfnick@polyu.edu.hk

Abstract

The flight delay-based operational aircraft maintenance routing problem (OAMRPFD) and the maintenance staffing problem (MSP) are interrelated and interdependent. Despite this interdependency, each problem is solved separately. Therefore, the optimal plan specified by each problem is not executed as planned, which consequently increases the operating cost of each aspect of the problem.

The present research paper studies OAMRPFD in addition to MSP, with two main objectives. The first is to develop an OAMRPFD model that appropriately reflects the flight delay and, accordingly, a new stochastic framework of a scenario-based OAMRPFD (SOAMRPFD) is put forward. The second is to handle the interdependence between SOAMRPFD and MSP, by proposing a coordinated configuration of SOAMRPFD and MSP that is formulated like the leader-follower Stackelberg game, with SOAMRPFD as the leader and MSP as the follower. A bi-level optimization model is used to present this game and is solved by a bi-level nested ant colony optimization (ACO) algorithm. The viability and potential of the proposed model are established using a major airline and a maintenance company, which are based in the Middle East, as a case study. The results demonstrate significant cost savings for both the airline and maintenance company.

Keywords: Aircraft maintenance routing problem, Maintenance staffing problem, Stackelberg game, Bi-level optimization.

1. Introduction

Airlines have used the aircraft maintenance routing problem (AMRP) as an efficient method for determining feasible aircraft maintenance routes (Haouari et al., 2012). Nevertheless, some challenges, due to increased flight delays, have appeared in the last decade. As a result, airlines have incurred increasing operating cost due to their inability to operate the generated aircraft routes as planned. For instance, Liang et al. (2015) stated that the operating cost of the U.S. airline industry in 2011 increased to about US$7.7 billion due to over 100 million minutes of delay. In the coming years, air traffic is expected to increase dramatically, hence, flight delays will be increased leading to serious revenue losses in the airline industry.

Accordingly, there is a paradigm shift from profit maximization to the minimization of the expected cost of flight delays in the airline industry (Dunbar et al., 2012; Lan et al., 2006; Liang et al., 2015).

AMRP has received tremendous research attention, with scholars focusing on three main areas: tactical (TAMRP), operational (OAMRP), and flight delay-based operational (OAMRPFD). TAMRP studies focus on the generation of typical rotations, which are repeated for each aircraft, while overlooking some operational constraints related to the maintenance process (Clarke et al., 1997; Gopalan & Talluri, 1998). As a result of ignoring these constraints, the generated routes could not be operated in reality, therefore, researchers developed OAMRP, which considers the operational constraints related to the maintenance process (Başdere & Bilge, 2014; Haouari et al., 2012). However, the main weakness of OAMRP is that it overlooks flight delays that are frequently encountered in reality, hence, the generated routes are prone to disruption due to delays. Thus, researchers have combined flight delays with the operational side, to generate realistic routes that can better withstand disruption (Lan et al., 2006; Liang et al., 2015). OAMRPFD studies are anchored on the expected value of a non-propagated delay. It should be noted that a non-propagated delay results from non-routing problems including bad weather, delays due to passengers, and technical problems. Nevertheless, the disadvantage of OAMRPFD lies in that the realized delays for some flights turn out to be significantly different from their expected values, due to the associated high degree of uncertainty (Yen & Birge, 2006). Consequently, flight delays are propagated, and the airlines incur increased related cost.

The use of OAMRPFD provides a solution (known as the routing plan), which involves assigning sequential flight legs to each aircraft, that includes some maintenance visits that satisfy the operational requirements. In particular, each aircraft starts covering the assigned flight legs, and then moves to the maintenance station. On completion of the maintenance process, the aircraft resumes covering its assigned flights. To achieve this solution as planned, the airline is required to send the scheduled times of arrival and departure for its aircraft to the maintenance company. Based on the received scheduled times, the maintenance company has to solve the maintenance staffing problem (MSP), so that it can determine the team sizes required for the arriving aircraft, while taking into account the workforce capacity and the scheduled times of the aircraft (Yan et al., 2004; Yang et al., 2003).

From the above, in achieving the solution of OAMRPFD as planned, it is the responsibility of the maintenance company to release the aircraft punctually from the maintenance station. Similarly, the MSP solution (known as the staffing plan) can be realized as planned if the airline sends their aircraft on time, so that any disturbance to the staffing plan can be avoided. Therefore, OAMRPFD and MSP are closely interdependent, and contingent on joint decision making.

Here, we study OAMRPFD along with MSP, with two main objectives. Firstly, to develop an OAMRPFD that appropriately reflects non-propagated delays. To achieve this, a new stochastic framework of a scenario-based OAMRPFD (SOAMRPFD) is proposed. Secondly, to examine the inherent interdependence between the proposed SOAMRPFD and MSP through a proposed coordinated configuration of SOAMRPFD and MSP that is formulated using the Stackelberg game, wherein SOAMRPFD and MSP act as a leader and a follower, respectively. This formulated leader-follower Stackelberg game can be modeled as a bi-level optimization model so that the existing coordination between SOAMRPFD and MSP can be captured. Furthermore, we propose a bi-level nested ACO algorithm as a solution method for the proposed model. The viability of the proposed model is established using a major airline and a maintenance company, which are based in the Middle East, as a case study.

The remainder of this paper proceeds as follows. In Section 2, we begin by discussing the literature review, research gaps and contribution of this study. Section 3 defines the problem context of SOAMRPFD, MSP, and the coordinated configuration. The upper-level optimization model (SOAMRPFD) is explained in Section 4, whereas the lower-level optimization model (MSP) is elaborated in Section 5. The bi-level optimization model is presented in Section 6. A bi-level nested ACO algorithm as a solution method is explained in Section 7. Using a major airline and a maintenance company as a case study, the performance analysis of the proposed model is presented in Section 8. Section 9 contains the conclusions of this study.

2. Literature review, research gap and contribution

2.1 Literature review

2.1.1 Aircraft maintenance routing problem

AMRP is handled by airlines focusing on three areas: tactical (TAMRP), operational (OAMRP), and flight delay-based operational (OAMRPFD). TAMRP studies focus on the generation of typical rotations, which are repeated for each aircraft, while overlooking some operational constraints related to the maintenance process. The OAMRP, on the other hand, aims to specify each aircraft's route while considering operational maintenance constraints. These constraints, as required by the Federal Aviation Administration (FAA), include restrictions on the total cumulative flying time, the total number of days between consecutive maintenance processes, and the total number of take-offs. Lastly, for OAMRPFD, all the operational restrictions must be obeyed besides considering fight delays, which frequently occur, and is neglected in the first two classifications.

A large number of studies with a tactical focus have been reported over last two decades. In the study by Kabbani and Patty (1992), a set-partitioning model was formulated for a 3-day AMRP so that lines of flight (LOF) could be achieved. Gopalan and Talluri (1998) expanded the use of LOF to cover AMRP for multiple days by developing a polynomial time algorithm, which was utilized to establish aircraft maintenance routes for a 3-day AMRP. Talluri (1998) developed an effective heuristic to solve the 4-day AMRP, which was shown to be NP-hard. With the aim of achieving feasible rotations that provide maximum through-value, Clarke et al. (1997) applied the lagrangian relaxation approach. The through-value is described as the extra profits derived from connecting flights with short transit times. A more recent study by Liang et al. (2011) focused on daily AMRP by proposing an integer linear programming (ILP) approach to capture their developed novel time-space network. Solutions of the proposed model were derived from a commercial software, called CPLEX. The application of tactical side studies, that uses a single rotation for each aircraft, is limited because operational constraints related to maintenance process are not taken into consideration. Therefore, the research focus has shifted from the tactical side to the operational side of AMRP.

Based on operational side of AMRP, Sriram and Haghani (2003) considered only the restriction on the total number of allowable days between consecutive maintenance processes, when they proposed their ILP model. Although the authors attempted to extend their model to consider the total cumulative flying time restriction, no solution was provided owing to the high complexity of the model. Further, Sarac et al. (2006) presented a set-partitioning formulation for OAMRP by considering a single maintenance constraint

(restriction on the total cumulative flying time), while ignoring others. Since the set-partitioning formulation produces an exponential number of feasible routes, the authors adopted column generation as a solution technique. A more improved non-linear OAMRP model, that simultaneously considered all three maintenance constraints, was reported by Haouari et al. (2012). They achieved linearization through a reformulation-linearization technique so that the proposed model could be easily solved using CPLEX. In another study, Başdere and Bilge (2014) considered the restriction on the total cumulative flying time as the only maintenance constraint and developed an ILP model for OAMRP. The authors utilized an exact method called branch-and-bound (B&B), but it could not solve large-scale problems. Therefore, to handle large scale problems, the authors adopted a metaheuristic method named compressed annealing (CA). The results revealed that CA was superior to B&B considering the relatively short computational time required for achieving feasible solution and its applicability to large-scale problems. Another OAMRP study was reported by Eltoukhy et al. (2017b) who proposed a model in which the restriction on the total cumulative flying time was considered as a maintenance constraint. In a follow-up paper, Eltoukhy et al. (2018) extended their previous work by proposing a polynomial formulation for OAMRP such that the main three maintenance constraints were considered. The authors solved their model by developing an efficient algorithm based on forward and backward insertion approaches. Although considerable research efforts have been expended in both TAMRP and OAMRP studies, the main weakness of these studies is that they ignore flight delays, which occur frequently in the real world. Consequently, the routes that are generated are readily disrupted by the delays and are difficult to implement in reality. For this reason, researchers now consider flight delays, besides the operational side, to produce routes that better withstand disruptions.

Few studies have been reported on the focus of OAMRPFD. The first study related to OAMRPFD was conducted by Lan et al. (2006). Their proposed model, whose objective was to reduce the total expected propagated delay, was solved using a column generation algorithm. Later, Dunbar et al. (2014) incorporated delay information in their model and proposed two algorithms to minimize the total delay propagation by re-timing the routes. The practical viability of the model was tested, and the computational results showed an average reduction of about 14% on the delay propagation. Recently, Liang et al. (2015) developed a new weekly LOF network model with the objective of minimizing the expected propagated delay and proposed a two-stage column generation approach for the solution. For a recent survey regarding AMRP, we refer interested readers to the review paper by Eltoukhy et al. (2017a).

2.1.2 Maintenance staffing problem

MSP is a problem that is handled by maintenance companies. On the focus of MSP, Dietz and Rosenshine (1997) proposed an approach that aimed at determining the optimal arrangement of workforce teams that provided maintenance for military aircraft. Beaumont (1997) also presented a model that was formulated as a mixed integer programming (MIP), so that the manpower supply could be minimized. With the same objective, Yang et al. (2003) proposed an MIP model that was used for deciding the size of work teams, the number of day shifts, and their starting time. Their proposed model took into account different flexible management strategies for efficiently managing the labor supply. In a follow up paper, Yan et al. (2004) expanded their previous work by considering some certification constraints, including the degree of training and functional abilities. They validated the model by using a case study from a Taiwan airline.

Although the MSP is usually solved separately, and its results used as an input to the rostering problem, there has been some research that considered both problem simultaneously (Beli et al., 2012; Beliën et al., 2013).

2.1.3 Leader-follower Stackelberg game

The basic form of the leader-follower decision structure coincides with the Stackelberg game (Stackelberg, 1952). This form of game includes two-level optimization problems; the upper-level (the leader) and the lower-level (the follower). In this hierarchical structure, the leader makes decisions firstly, due to holding a powerful position, and announces these decisions to the follower. Then, the follower reacts rationally to the received decisions, through making the decisions and feeding them back to the leader. The application of the leader-follower Stackelberg game has been successfully demonstrated in different fields, including advertising (Aust & Buscher, 2012), the manufacturer-distributor supply chain (Qin, 2012), the retailer-manufacturer supply chain (Tsao et al., 2014), pricing (van Hoesel, 2008), and the seller-buyer supply chain (Esmaeili et al., 2009; Xiao et al., 2014).

2.2 Research gap

From the above literature review, we can see that most researchers handled OAMRPFD and MSP in isolation or separately, neglecting the interdependence between them. From a practical point of view, airlines and maintenance companies have a close relationship. To airlines, a flight will be delayed if an aircraft cannot be released from the maintenance station punctually. Similarly, for maintenance companies, if an aircraft misses the scheduled appointment at the maintenance station, this will also cause a huge disruption to the maintenance company's original plan. So, if these two problems are solved separately, the optimal aircraft routing plan determined by OAMRPFD may not be fulfilled because the planned arrival and departure times cannot be satisfied due to constraints in the MSP. In a similar way, the optimal staffing plan determined by MSP may not be realized because of constraints in OAMRPFD. This situation will result in severe flight delays, and consequently lead to a waste of resources and energy.

To our best knowledge, so far, in the existing research, there has been no attempt to investigate the interdependence between OAMRPFD and MSP. In this connection, this paper attempts to fill this gap by providing a coordinated configuration between OAMRPFD and MSP through the adoption of the leader-follower Stackelberg game. In this sense, this research is a step closer to the real situation as compared to the research work found in the literature.

2.3 Contribution

This paper is focused on OAMRPFD along with MSP and has two contributions. Firstly, it is observed in the reviewed studies that the formulation of OAMRPFD was premised on the expected value of the non-propagated delay, defined as delays initiated by bad weather, technical problems, delays due to passengers, etc., which are classified as non-routing issues. Nevertheless, the disadvantage of this formulation is manifested as a significant difference between the realized delay (RD) for some flights and their expected values (EV), owing to the high level of uncertainty associated with the non-propagated delays (Yen & Birge, 2006). To provide an empirical evidence for the significant difference between the RD for some flights and their EV, we conduct a computational experiment using the data acquired from a Middle Eastern major airline. This experiment includes three steps. First step is collecting the flight information for all

flights recorded by the airline from January 2016 to December 2016, including flight number, departure airport, departure time, arrival airport, arrival time, visited maintenance station, day of operation, and flight duration. The data contain a total of 292,000 flights flown by 12 fleets. Second step is analyzing the collected data and selecting some fleets for our experiment. While analyzing the data, we consider any flight as a delayed flight even if its related non-propagated delay time is less than 15 minutes, as any non-propagated delay may easily cause the propagated delay in real practice. Meanwhile, any non-propagated delay time longer than 170 minutes is discarded because it reflects a severe disruption, which is not in the scope of this study. Based on this analysis, we pick the top three fleets that suffer from large number of delayed flights, as shown in Table 1. Last step of the experiment is using the collected data to calculate the EV of the non-propagated delay for the flights flown by the picked fleets, calculating the difference between the RD and the EV, and finally counting the number of flights with significant difference between RD and EV. These flights with significant difference are called FWSD. In this experiment, we consider any flight as a FWSD if the difference between RV and EV is larger than 45 minutes as recommended by experts in the airline. The results of the experiment are summarized in the last two columns of Table 1, in which the number of FWSD and the likelihood of facing these flights, are reported. The results of this experiment reveal that the number of FWSD is quite large, as being 13.58%, 11.28%, and 10.24% of total flights flown by first, second, and third fleet, respectively. A consequence of this situation is that the delay is propagated, and its related cost is increased. This forms the basis of our motivation to find a better alternative to capture the non-propagated delay. Therefore, we study different potential scenarios for the non-propagated delay, which enable us to derive a suitable look-ahead feature, as opposed to the expected value approach. Furthermore, this scenario-based concept results in proposing SOAMRPFD, which has been demonstrated to be a very successful method for handling stochastic parameters (Chen et al., 2002; Mohammadi et al., 2014; Samimi et al., 2017).

Table 1:Features of selected fleets

Fleet	Total flights	Delayed flights		FWSD	
		No.	%=No./Total flights	No.	%=No. /Total flights
A300	21,960	6,318	28.77	2,982	13.58
B757	29,280	6,204	21.19	3,303	11.28
B737	18,300	3,665	20.03	1,874	10.24

Secondly, from the literature review, one of the significant gaps is caused by solving OAMRPFD and MSP separately. However, the pitfall of this separation is ignoring the interdependence between them, as discussed earlier. In this paper, for the first time, we attempt to fulfill this research gap by solving the proposed SOAMRPFD and MSP, while considering their interdependence. To do so, it is much trickier to find an appropriate approach, while considering the different characteristics of each problem. Firstly, SOAMRPFD and MSP are of different companies with inconsistent objectives. This inconsistency stems from the fact that SOAMRPFD seeks to minimize the expected propagated delay cost, which requires maximizing the worker team sizes in the maintenance company in order to release the aircraft punctually, resulting in an increase in the labor cost. On the other hand, MSP aims at minimization of the labor cost, which might result in releasing the aircraft behind the scheduled departure time, leading to an increase in the propagated delay cost. Therefore, handling both problems using “all-in-one” approach, which enables the combination of the two objectives into a single objective function is not viable. In this sense, handling both problems appropriately in the presence of conflicting goals as well as the interdependence makes the

coordinated configuration between them indispensable. Practically, SOAMRPFD determines the number of aircraft to be maintained, which constitutes the demand for the maintenance company. Based on the demand as an input, MSP is solved, and the staffing plan is determined. From this description, we can see that SOAMRPFD holds a powerful position as it determines the demand, and MSP acts based on the received demand. This situation leads naturally to the formulation of the coordinated configuration of SOAMRPFD and MSP as a leader-follower Stackelberg game, with the SOAMRPFD acting as a leader, and MSP as a follower, responds rationally to the SOAMRPFD decisions. This game can be represented using a bi-level optimization model that captures the coordination among both parties. Using such a coordinated configuration is helpful to the airlines and maintenance companies for reducing resource wastage and energy consumption caused by flight delays or disruptions, and in improving their service levels.

3. Problem description

3.1 Stochastic operational aircraft maintenance routing problem

SOAMRPFD is regarded as the first part of the coordinated configuration, which is handled by the airlines. The traditional task of SOAMRPFD is to generate feasible maintenance routes for each aircraft flown. A feasible maintenance route is defined as that which respects the operational maintenance restrictions mandated by FAA including the maximum number of allowable flying time, the maximum number of allowable days between consecutive maintenance processes, and the maximum number of allowable take-offs (Haouari et al., 2012). In order to generate these routes, SOAMRPFD has to deal with two main issues, as shown in the left-hand side of Fig.1. Firstly, the aircraft that has a limited number in the airline, and secondly, is the list of flight legs. Each flight leg is characterized by six features: departure time, origin airport, arrival time, destination airport, flight duration, and non-propagated delay. The critical issue of SOAMRPFD is to build routes while satisfying all the operational maintenance requirements. Meanwhile, it becomes more challenging to generate these routes when considering the non-propagated delay for each flight. It should be noted that handling the non-propagated delays is carried out by generating different disruption scenarios, resulting in more complexity being added to the problem.

Insert Fig. 1 appropriate here

Based on the above information and requirements, the solution of SOAMRPFD has the objective of minimizing the expected cost of the propagated delays. A flight delay in the airline industry can be categorized as a propagated delay and a non-propagated delay. Regarding the non-propagated delay, it is ascribed to bad weather, technical problems, passenger, etc., which are generalized as non-routing reasons. On the other hand, a propagated delay is described as when the aircraft covering a later flight is delayed due to a delay on its previous covered flight. Based on these definitions, we can see that the propagated delay is a function of routing decisions. Therefore, minimizing the expected cost of propagated delays is logically the main objective of SOAMRPFD.

SOAMRPFD provides a solution (known as the routing plan), which is illustrated on the right-hand side of Fig. 1. The generated solution consists of sequential flight legs assigned to each aircraft, with each sequence including some maintenance visits to satisfy the operational requirements. In particular, each aircraft starts covering the assigned flight legs as planned, then moves to the maintenance station for maintenance. It

should be pointed out here that there are number of maintenance stations located on the hub airports where the maintenance companies can work at. On completion of the maintenance, the aircraft moves from the maintenance station in order to resume covering its assigned flights. To achieve this goal, it is the responsibility of the maintenance company to release the aircraft punctually from the maintenance station. This requires the maintenance company to assign their workforce to perform the maintenance processes efficiently by solving the MSP.

From the above description, it becomes clear that achieving solution of SOAMRPFD as planned is not only dependent on the airlines, but also dependent on the maintenance companies, as their operations are part of the solution. Therefore, an essential requirement is to optimize SOAMRPFD along with MSP decisions. In other words, SOAMRPFD makes decisions, including the scheduled arrival and departure times of each aircraft at maintenance stations, and sends these decisions to the maintenance companies. By observing the sent decisions, MSP makes decisions, including the required team sizes and the actual departure time for the aircraft from the maintenance station. Then, MSP decisions, especially the actual departure time after receiving the maintenance process, are sent back to the aircraft company. If these decisions affect the initial constructed routes, the SOAMRPFD is resolved in order to improve the obtained solutions.

3.2 Maintenance staffing problem

The second part of the coordinated configuration is represented by MSP, which is solved by the maintenance companies that own maintenance stations located on the hub airports where the maintenance companies can work at. The main goal of MSP is to determine the workforce team sizes required to perform the maintenance processes to the aircraft. To do so, MSP has to manage three main issues: the received aircraft, the workforce, and the planned shifts, as illustrated in the left-hand side of Fig. 2. Starting with the aircraft, it represents the demand for the maintenance company, which is characterized by the scheduled arrival and departure times for each aircraft. Moving to the workforce, each maintenance company has its own workforce capacity and limits on team sizes. Lastly, for planned shifts, some companies build their plan based on a two-shift pattern (morning and afternoon), whereas others deal with a three-shift pattern (morning, afternoon, and night). The critical issue of MSP is to determine efficient team sizes while taking into account the workforce capacity and the scheduled arrival and departure times of the aircraft. Clearly, when the team size is large, the maintenance time will be shorter, resulting in an early departure for the aircraft from the station ahead the scheduled departure time, and vice versa.

Insert Fig. 2 appropriate here

Using the aforementioned information, the main objective of solving MSP is to minimize the labor cost. The generated solution from MSP (known as the staffing plan) indicates how many workers are assigned to maintain each aircraft, and accordingly, the actual departure time for each aircraft from the maintenance station can be determined. Sometimes, due to workforce capacity restrictions, the solution in terms of actual departure time might be behind the scheduled departure time, resulting in a delay to the aircraft. As a result, the routing plan determined by the airline will be affected, and its plan will not be realized. Meanwhile, if the aircraft misses its scheduled appointment at the maintenance station, this will also cause a huge interruption to maintenance company's staffing plan. So, it is again confirmed from the MSP side that achieving the staffing plan is a mutual responsibility between the airline and maintenance companies. Therefore, MSP should be solved jointly with SOAMRPFD, in which the MSP decisions including the

actual departure time for the aircraft are sent back to the airline. In a response to these decisions, the airline will change its decisions, including the scheduled arrival and departure time, if its initial plan is affected.

3.3 Coordinated configuration

Based on the above description, both SOAMRPFD and MSP have inconsistent objectives. This inconsistency stems from the fact that SOAMRPFD seeks to minimize the expected propagated delay cost, which requires maximizing the worker team sizes in the maintenance company in order to release the aircraft punctually, resulting in an increase in the labor cost. On the other hand, MSP aims at minimization the labor cost, which might result in releasing the aircraft behind the scheduled departure time, leading to an increase in the propagated delay cost. These conflicting goals as well as the interdependence between SOAMRPFD and MSP make the coordinated configuration between them indispensable. One of the tools that can handle this situation is the Stackelberg game, known as leader-follower game. This game can be adopted to handle decisions of two self-interested problems with hierarchical features. In this game, SOAMRPFD firstly makes decisions in terms of the routing plan and then MSP responds to SOAMRPFD decisions by making decisions in terms of the staffing plan. As a result, the coordinated configuration between SOAMRPFD and MSP enables the formulation of the problem as a leader-follower Stackelberg game, where SOAMRPFD acts as a leader and MSP as a follower, as shown in Fig. 3.

Insert Fig. 3 appropriate here

Fig. 3 elaborates such interdependence between two decision makers in the coordinated configuration, whereas Table 2 summarizes the definitions of the sets, parameters and decision variables used throughout SOAMRPFD and MSP models. The upper part of Fig. 3 represents the airline and its SOAMRPFD, whereas the lower part represents the maintenance companies and their MSP. Acting as leader, SOAMRPFD makes decisions according to the routing plan ($ATBM_{kvm}^{\xi}$ and $RTAM_{kvm}^{*\xi}$), then these decisions are sent to the maintenance companies. Based on the received decisions, the maintenance companies calculate the scheduled arrival and departure times for the aircraft, known as SAT_{kvm}^{ξ} and SDT_{kvm}^{ξ}, respectively. Based on SAT_{kvm}^{ξ} and SDT_{kvm}^{ξ} as the input, MSP makes decisions according to the staffing plan (wf_{kvsm}^{ξ} and $RTAM_{kvm}^{\xi}$). Then, the MSP decisions are sent back to SOAMRPFD. If these decisions affect the initial routing plan, SOAMRPFD is resolved. This process is iterated until reaching the Stackelberg equilibrium, in which both leader and follower do not have any intention to adjust their decisions, as any change will result in negative impact in their objective functions. Accordingly, the optimal setting for the coordinated configuration can be derived.

Table 2: Sets, parameters and decision variables.

Upper- level optimization model (SOAMRPFD decision model)	
Sets	
$i, j \in I$:	Set of flight legs.
$m \in MT$:	Set of maintenance stations.
$k \in K$:	Set of aircraft.
$a \in A$:	Set of airports.

$\xi \in \Psi$:	Set of scenarios corresponding to non-propagated delay realizations (known as disruption scenarios).
$v \in \{1,2,\dots,\Omega\}$	Average number of maintenance processes that each aircraft should receive during the planning horizon.
$\{o,t\}$:	Artificial source and sink nodes of the aircraft routing network.
Parameters	
DT_i:	Local time when flight leg i departs from its origin airport, known as departure time.
O_{ia}:	Binary indicator for the origin airport of flight leg i. It equals 1 when the origin airport of flight leg i and the airport a are the same, and 0 otherwise.
AT_i:	Local time when flight leg i arrives at its destination airport, known as arrival time.
D_{ia}:	Binary indicator for the destination airport of flight leg i. It equals 1 when the destination airport of flight leg i and the airport a are the same, and 0 otherwise.
FT_i:	Duration of flight leg i.
TRT:	Time consumed for getting passengers off, unloading the luggage, changing the gate, boarding, loading the luggage, moving the aircraft between the gate and maintenance station, and fueling the aircraft. It is Known as turn-around time.
T_{max}:	Maximum number of allowable flying time since last maintenance process.
C_{max}:	Maximum number of allowable take-offs since last maintenance process.
NPD_i^{ξ}:	Realized value of the non-propagated delay of flight leg i, while considering disruption scenario ξ.
Mb_{ma}:	Binary indicator for maintenance station m. It equals 1 when the location of the maintenance station m and the airport a are identical, and 0 otherwise.
MAT:	Duration of Type A maintenance check. The airline assumes this value only in the first round of the coordination between the airline and maintenance companies.
KT:	Fleet size.
Ω:	Maximum average number of maintenance processes that each aircraft should receive during the planning horizon. It is calculated according to the following equation; $\Omega = \sum_{i \in I} FT_i/(T_{max}\ KT)$
M:	A big number.
PD_{ijkv}^{ξ}:	Value of propagated delay resulted when aircraft k covers two successive flight legs i and j, before receiving the maintenance process number v, while considering disruption scenario ξ.
PD_{ikv}^{ξ}:	Value of accumulated propagated delay resulted after covering flight leg i by aircraft k, before receiving maintenance process number v, while considering disruption scenario ξ. It should be pointed out here that this propagated delay is the total delay, due to non-propagated delays that are

	ascribed to bad weather, technical problems, passenger, and congestion of maintenance stations.
p^{ξ}:	Probability in which disruption scenario ξ can be realized.
C_{pD}:	Expected cost per each minute of propagated delay.
Decision variables	
$x_{ijkv}^{\xi} \in \{0,1\}$:	Flight coverage decision variable, while considering disruption scenario ξ. x_{ijkv}^{ξ}=1 when aircraft k covers two successive flight legs i and j, before receiving the maintenance process number v and 0 otherwise.
$y_{imkv}^{\xi} \in \{0,1\}$:	Visiting maintenance station decision variable, while considering disruption scenario ξ. y_{imkv}^{ξ}=1 when flight leg i is covered by aircraft k then the aircraft proceeds to maintenance station m to receive the maintenance process number v and 0 otherwise.
$z_{mjkv}^{\xi} \in \{0,1\}$:	Leaving maintenance station decision variable, while considering disruption scenario ξ. z_{mjkv}^{ξ}=1 when aircraft k leaves maintenance station m in order to cover flight leg j, after receiving the maintenance process number v and 0 otherwise.
$ATBM_{kvm}^{\xi} > 0$	Scheduled arrival time of aircraft k at maintenance station m to receive the maintenance process number v, while considering disruption scenario ξ.
$RTAM_{kvm}^{*\xi} > 0$:	Scheduled departure time of the aircraft k from maintenance station m, after receiving the maintenance process number v, while considering disruption scenario ξ.
<u>Lower-level optimization model (MSP decision model)</u>	
Sets	
$kv \in \{11, \dots, 1\Omega, \dots bv, \dots, K\Omega\}$:	Set of aircraft that are planned to visit the maintenance station in order to receive a specific number of maintenance process. Each element in this set consists of two parts such that the first part, k, represents the aircraft, whereas the second part, v, represents the number of the maintenance process that will be received by the aircraft. For example, 11, refers to the aircraft number 1 that is planned to receive maintenance process number 1.
$s \in S$:	Set of shifts.
$\{o', t'\}$:	Artificial source and sink nodes of the layered graph.
Parameters	
SAT_{kvm}^{ξ}:	Scheduled arrival time of an aircraft k to receive maintenance process number v, at maintenance station m, while considering disruption scenario ξ received from airline.
SDT_{kvm}^{ξ}:	Scheduled departure time of an aircraft k after receiving maintenance process number v, at maintenance station m, while considering disruption scenario ξ received from airline.

w_{sm}^{l}:	Minimal team size (number of workers) that can be formed to perform a maintenance process, during shift s, at maintenance station m.
w_{sm}^{u}:	Maximal team size (number of workers) that can be formed to maintain an aircraft, during shift s, at maintenance station m.
Q_{s}^{max}:	Capacity of workforce available during shift s.
l_{kv}:	Workload (man-hours) required to maintain an aircraft k that receives maintenance process number v.
C_{wkvsm}:	Cost incurred when assigning w workers to maintain an aircraft k that receives maintenance process number v, during shift s, at maintenance station m. Suppose that the maintenance companies having certain personnel and it is probably no ability to change the employment level over the planning horizon; how can the cost incurred be variable. To answer this question, it is should be pointed out that the payment system for the workers in the Middle Eastern maintenance companies is based on the type of the shift. In real practice, most of the workers prefer to serve at the morning and the afternoon shifts instead of serving at the night shift. Therefore, the cost incurred for the team served at the morning and the afternoon shifts is relatively lower than the night shift. Consequently, the cost incurred might be variable for the same personnel over the planning horizon of the model, depending on the shifts when the workers serve the aircraft. Note that these shifts are determined according to the arrival and departure times of the aircraft at maintenance stations.
Decision variables	
$wf_{kvsm}^{\xi} \in \{w_{sm}^{l}, \dots, w_{sm}^{u}\}$:	Number of workers (team size) assigned to maintain an aircraft k that receives maintenance process number v, during shift s, at maintenance station m, while considering disruption scenario ξ.
$RTAM_{kvm}^{\xi} > 0$:	Actual departure time of an aircraft k after receiving maintenance process number v, at the maintenance station m, while considering disruption scenario ξ.

4. Upper-level optimization model (SOAMRPFD decision model)

With the objective of minimizing the total expected propagated delay cost for all aircraft, SOAMRPFD is required to establish feasible maintenance routes for each aircraft. The formulation of SOAMRPFD is anchored on the connection network due to its efficiency in capturing the whole image of the routing problem (Gopalan & Talluri, 1998; Haouari et al., 2012). The nodes in the original connection network correspond to the flight legs, while the arcs denote the potential connections between those flight legs. As mentioned earlier, our proposed SOAMRPFD model considers two main issues. Firstly, considering three main maintenance constraints. Secondly, determining the appropriate time and location each aircraft should undergo maintenance processes. As mentioned earlier, these maintenance processes are performed at a number of maintenance stations located on the hub airports where the maintenance companies can work at. The previous two considerations can be done as follows. Initially, we determine the average number of maintenance processes that should be received by each aircraft, based on the overall number of flying hours in the schedule, the available number of aircraft, and the maximum number of allowable flying time for

each aircraft. Next, the number of maintenance processes is assigned to each aircraft in the fleet. In order to keep covering the flight legs and assigning maintenance processes simultaneously, we need to add other arc and node types to the original connection network. Therefore, it is slightly modified by adding one type of nodes (known as the maintenance station node) and two types of arcs (known as maintenance arc and auxiliary arc), as shown in Fig. 4. The node set includes the maintenance station node set (MT) besides the flight leg node set (I). On the other hand, the arc set is modified to include the maintenance arc set ($MAINT$) and the auxiliary arc set (AUX), besides the ordinary arc set (ORD). In our modified network, the ordinary arc $ord(i,j) \in ORD$ is used for connecting between flight legs i and j, between the flight leg and source node at the beginning of the route construction, and between the flight leg and sink node at the end of the route construction. On the other hand, the maintenance arc $maint(i,m) \in MAINT$ is incorporated in the network for connecting between flight leg i and maintenance station m, whereas the auxiliary arc $aux(m,j) \in AUX$ is added for connecting between maintenance station m and flight leg j. Finally, the auxiliary arc helps the aircraft to resume covering the flight legs after finishing the maintenance processes.

Insert Fig. 4 appropriate here

Modifying the connection network helps us to formulate SOAMRPFD as a multi-commodity network flow model, such that each aircraft represents a separate commodity circulating in the network. Definitions of the sets, parameters, and decision variables are summarized in Table 2.

The used decision variables are x_{ijkv}^{ξ},y_{imkv}^{ξ} and z_{mjkv}^{ξ}, which represent ordinary arcs, maintenance arcs, and auxiliary arcs, respectively. In addition, decision variables $ATBM_{kvm}^{\xi}$and $RTAM_{kvm}^{*\xi}$are used in order to help in calculating the scheduled arrival and departure times for aircraft when visiting the maintenance stations. The non-propagated delays randomness in the proposed model is represented using a set of generated scenarios Ψ, in which each scenario ξ defines a set of possible non-propagated delay realizations for scheduled flight legs. Clearly, if, for instance, the number of scheduled flight legs is 240, then 240 non-propagated delay realizations represent a single scenario ξ, such that each flight leg takes a single non-propagated delay realization. By incorporating these scenarios in the proposed model, we can simulate the potential disruptions that may occur in the future, resulting in the generation of a routing plan that can be easily implemented in reality. These scenarios can be generated using the following procedures:

- Data collection. It should be noted that, if the scenarios will be generated for a group of flight legs, the data should be collected for these flights, named the target flights. For this purpose, we collect the flight information for the target flights flown by the airline during the last year, including flight number, departure airport, departure time, arrival airport, arrival time, flight duration, and the non-propagated delay for each flight.
- Data preprocessing. As mentioned earlier, for the collected data, any flight is considered as a delayed flight even if its related non-propagated delay time is less than 15 minutes. This because any non-propagated delay may easily cause a propagated delay in practice. In addition, any non-propagated delay time longer than 170 minutes is discarded, as it reflects a severe disruption, which is not in the scope of this study.
- Overall data analyzing. Using the collected data, determine the number of delayed flights and its percentage compared to the total flights. For each delayed flight, determine the value of the non-propagated delay and its related cause. In our preliminary experiments, we discovered three main

causes for the non-propagated delay; passengers delays, technical problems, and bad weather. Next, determine the percentage of each cause for the non-propagated delay among other causes.

- Specific data analyzing. For each flight leg in the target flight, determine the number of times such that the flight leg appears in the following situations; free of delay, delay due to passenger delays, delay due to technical problems, and delay due to bad weather. Next, determine the percentage of each situation. The main reason for adding this procedure is that, using the previous procedure, overall data analyzing, to generate the disruption scenarios results in producing scenarios that only respect the overall percentage of delayed flights and the overall percentage of each cause of non-propagated delay, but overlook the detailed history for each flight leg. This in turn results in generating a set of scenarios that do not represent the correct situations.
- Scenario generation. By considering the information calculated in the previous two procedures, generate the required number of scenarios using a truncated gamma distribution for the length of non-propagated delays to match delay data collected from the airline for mean, second moment, and range.

Before presenting the SOAMRPFD decision model, we specify its scope as follows:

- The planning horizon of the proposed model is 4-day. It should be noted that, in the literature, there are three main planning horizons; daily, 4-day, and weekly. The daily horizon assumes that the flight schedule is repeated every day of the week. On the other hand, 4-day or weekly horizons permit different flight schedules for each day of the week. Practically, using daily horizon is not viable as airlines permit variations on the flight schedule for each day of the week to cope with the demand fluctuation of different flight legs (e.g. the demand at weekend is higher than other days) (Eltoukhy et al., 2017a). In light of this fact, 4-day or weekly horizons are more practical in handling different flight schedules each day, so that the proposed model can handle real flight schedules. Extending the planning horizon to be 4-day or weekly has some advantages, as mentioned, but extensions to weekly horizons produce routes that are susceptible to disruptions. Therefore, 4-day is preferable for airlines compared to weekly horizon. In addition, airlines tend to use 4-day planning horizon in order to ease the requirement of satisfying one maintenance visit every 4 days for the aircraft, as mandated by FAA (Eltoukhy et al., 2018). Moreover, the 4-day horizon is commonly used in the literature (Feo & Bard, 1989; Talluri, 1998). Therefore, based on the previous observations, we stick with the 4-day horizon to be consistent with the existing research and practice.
- The hub airports host the maintenance stations.
- The decision model is limited to the existing maintenance stations and does not consider the construction of new ones
- The decision model only considers Type A maintenance checks, as frequently reported in past publications.

Based on the predefined scope and notations, the optimization model for SOAMRPFD, as a leader, can formulated as follows:

$$min \sum_{\xi \in \Psi} p^{\xi} \left\{ \sum_{v=1,\dots,\Omega} C_{pD} \left(\sum_{k \in K} \sum_{i \in I} \sum_{j \in I} PD_{ijkv}^{\xi} \, x_{ijkv}^{\xi} \right) \right\} \tag{1.0}$$

$$\text{s.t.} \quad PD_{ijkv}^{\xi} = PD_{ikv}^{\xi} + (NPD_{i}^{\xi} - (DT_j - AT_i - TRT))^{+} \qquad \forall (i,j) \in I, k \in K, v = 1, \dots, \Omega, \xi \in \Psi \tag{1.1}$$

$$\sum_{k\in K}\left(\sum_{j\in I\cup\{t\}}\sum_{v=1,\dots,\Omega}x_{ijkv}^{\xi}+\sum_{m\in MT}\sum_{v=1,\dots,\Omega}y_{imkv}^{\xi}\right)=1 \qquad \forall\, i\in I,\xi\in\Psi \tag{1.2}$$

$$\sum_{j\in I}x_{ojkv}^{\xi}+\sum_{m\in MT}y_{omkv}^{\xi}=1 \qquad \forall\, k\in K,v=1,\xi\in\Psi \tag{1.3}$$

$$\sum_{i\in I}x_{itkv}^{\xi}+\sum_{m\in MT}z_{mtkv}^{\xi}=1 \qquad \forall\, k\in K,v=\Omega,\xi\in\Psi \tag{1.4}$$

$$\sum_{j\in I\cup\{o\}}x_{jikv}^{\xi}+\sum_{m\in MT}z_{mikv}^{\xi}=\sum_{j\in I\cup\{t\}}x_{ijkv}^{\xi}+\sum_{m\in MT}y_{imkv}^{\xi} \qquad \forall\, i\in I,k\in K,v=1,\dots,\Omega,\xi\in\Psi \tag{1.5}$$

$$\sum_{j\in I}\sum_{v=1,\dots,\Omega}y_{jmkv}^{\xi}=\sum_{j\in I\cup\{t\}}\sum_{v=1,\dots,\Omega}z_{mjkv}^{\xi} \qquad \forall\, m\in MT,k\in K,\xi\in\Psi \tag{1.6}$$

$$AT_i+TRT-DT_j\le M(1-x_{ijkv}^{\xi}) \qquad \forall\,(i,j)\in I,k\in K,v=1,\dots,\Omega,\xi\in\Psi \tag{1.7}$$

$$\sum_{k\in K}x_{ijkv}^{\xi}\le\sum_{a\in A}D_{ia}O_{ja} \qquad \forall\,(i,j)\in I,v=1,\dots,\Omega,\xi\in\Psi \tag{1.8}$$

$$\sum_{k\in K}y_{imkv}^{\xi}\le\sum_{a\in A}D_{ia}Mb_{ma} \qquad \forall\, i\in I,m\in MT,v=1,\dots,\Omega,\xi\in\Psi \tag{1.9}$$

$$ATBM_{kvm}^{\xi}\ge\sum_{i\in I\cup\{o\}}\sum_{m\in MT}\left(AT_i+TRT+PD_{ikv}^{\xi}\right)y_{imkv}^{\xi} \qquad \forall\, k\in K,v=1,\dots,\Omega,\xi\in\Psi \tag{1.10}$$

$$\sum_{k\in K}z_{mjkv}^{\xi}\le\sum_{a\in A}Mb_{ma}\,O_{ja} \qquad \forall\, m\in MT,j\in I,v=1,\dots,\Omega,\xi\in\Psi \tag{1.11}$$

$$RTAM_{kvm}^{*\xi}-DT_j\le M\left(1-z_{mjkv}^{\xi}\right) \qquad \forall\, m\in MT,j\in I,k\in K,v=1,\dots,\Omega,\xi\in\Psi \tag{1.12}$$

$$RTAM_{kvm}^{*\xi}\ge\sum_{i\in I\cup\{o\}}\sum_{m\in MT}\left(AT_i+TRT+PD_{ikv}^{\xi}+MAT\right)y_{imkv}^{\xi} \qquad \forall\, k\in K,v=1,\dots,\Omega,\xi\in\Psi \tag{1.13}$$

$$RTAM_{kvm}^{*\xi}\ge RTAM_{kvm}^{\xi} \qquad \forall\, k\in K,v=1,\dots,\Omega,m\in MT,\xi\in\Psi \tag{1.14}$$

$$\sum_{i\in I\cup\{o\}}\sum_{j\in I}x_{ijkv}^{\xi}\le C_{max} \qquad \forall\, k\in K,v=1,\dots,\Omega,\xi\in\Psi \tag{1.15}$$

$$\sum_{i\in I\cup\{o\}}\sum_{j\in I}FT_j\;x_{ijkv}^{\xi}\le T_{max} \qquad \forall\, k\in K,v=1,\xi\in\Psi \tag{1.16}$$

$$\sum_{i\in I}\sum_{j\in I}FT_j\,x_{ijkv}^{\xi}+\sum_{m\in MT}\sum_{j\in I}FT_j\,z_{mjkv}^{\xi}\le T_{max} \qquad \forall\, k\in K,v=2,\dots,\Omega,\xi\in\Psi \tag{1.17}$$

$$\sum_{i\in I}\sum_{m\in MT}\sum_{v=1,\dots,\Omega}y_{imkv}^{\xi}\ge 1 \qquad \forall\, k\in K,\xi\in\Psi \tag{1.18}$$

$$x_{ijkv}^{\xi}\in\{0,1\} \qquad \forall\;(i,j)\in I,k\in K,v=1,\dots,\Omega,\xi\in\Psi \tag{1.19}$$

$$y_{imkv}^{\xi}\in\{0,1\} \qquad \forall\, i\in I,m\in MT,k\in K,v=1,\dots,\Omega,\xi\in\Psi \tag{1.20}$$

$$z_{mjkv}^{\xi}\in\{0,1\} \qquad \forall\, m\in MT,\; j\in I,k\in K,v=1,\dots,\Omega,\xi\in\Psi \tag{1.21}$$

$$ATBM_{kvm}^{\xi}>0 \qquad \forall\, k\in K,v=1,\dots,\Omega,m\in MT,\xi\in\Psi \tag{1.22}$$

$$RTAM_{kvm}^{*\xi}>0 \qquad \forall\, k\in K,v=1,\dots,\Omega,m\in MT,\xi\in\Psi \tag{1.23}$$

The objective function (1.0) is required to minimize the total expected cost of the propagated delay, under all generated disruption scenarios. Constraints (1.1) describe the calculation of the propagated delay caused when an aircraft will potentially cover flight leg j after covering flight leg i. By looking precisely at constraints (1.1), we can notice that they are sequence-dependent constraints. This is because the propagated

delay when covering flight j, represented by PD_{ijkv}^{ξ}, is dependent on the accumulated propagated delay caused at its previous flight leg i, known as PD_{ikv}^{ξ}.

Constraints (1.2), (1.3), and (1.4) are cast, which ensure that all the flight legs are covered. Constraints (1.2) ensure the exact coverage of each flight leg by a single aircraft. Constraints (1.3) and (1.4) ensure the route initiation and the route completion for each aircraft, respectively.

Some balance constraints (1.5) and (1.6) are designed to maintain the aircraft movement throughout the connection network. The constraints in (1.5) ensure that a balance is maintained for aircraft during the process of covering the flight leg nodes. In other words, they ensure that when the aircraft uses the ordinary or the auxiliary arc to cover a flight leg, it is necessary for the aircraft to use the ordinary or the maintenance arc to cover the subsequent flight leg. Similar to constraints (1.5), constraints (1.6) keep a balance for aircraft during the process of visiting the maintenance stations. Constraints (1.6) ensure that if the aircraft covers the flight leg and visits the maintenance station by usage of the maintenance arc, then the aircraft must leave the maintenance station by using the auxiliary arc to cover the next flight.

To let an aircraft to cover two consecutive flight legs by adoption of the ordinary arc, it is important to consider the time and place issues of these two flight legs. Therefore, constraints (1.7) and (1.8) are cast. The time constraints indicated in (1.7) guarantee the existence of the turn-around time between the arrival and departure time of two successive flight legs, when these flight legs are covered by the same aircraft. The place constraints in (1.8) ensure that two successive flight legs can be flown by an aircraft on the condition that the same airport is shared between the destination airport and the origin airport of the first and the second flight leg, respectively.

One of the important aspects of SOAMRPFD is in allowing the aircraft to receive the maintenance process after covering a flight leg. This can be done by using the maintenance arc and taking cognizance of the location of the potentially visited maintenance station. This is summarized by constraints (1.9). They guarantee that a maintenance station can be visited by an aircraft, provided that the same location is shared by the destination airport of the last covered flight leg and the location of the maintenance station. In real practice, to receive the maintenance process, the airline should inform the maintenance station with the scheduled arrival time of the aircraft. For this purpose, constraints (1.10) are cast. These constraints ensure that the scheduled arrival time of the aircraft at the maintenance station is larger than or equal the arrival time of the last covered flight leg plus the turn-around time and the value of accumulated propagated delay.

On completion of the maintenance process, the aircraft is required to leave the maintenance station and use the auxiliary arc to proceed to the next flight leg. To achieve this, constraints (1.11), (1.12), (1.13) and (1.14), representing place and time constraints regarding this situation. The place constraints are indicated by constraints (1.11). They mandate that the aircraft is only allowed to cover the next flight leg after receiving the maintenance process on the condition that the same location is shared by the maintenance station and the origin airport of the next flight leg. On the other hand, the time constraints are summarized in (1.12). They ensure that the next flight leg can proceed by the aircraft on the condition that the scheduled departure time of aircraft $RTAM_{kvm}^{*\xi}$ determined by constraints (1.13) and (1.14), is less than or equal to the departure time of the next flight leg. Initially, in the first round of the coordination between SOAMRPFD and MSP, $RTAM_{kvm}^{*\xi}$ is determined by using constraints (1.13), which include an assumption made by the airline regarding the duration of maintenance process. In reality, this assumption is not applicable, because this duration is determined by the maintenance company not by the airline. Therefore, in the subsequent

rounds of coordination, constraints (1.13) become redundant, and $RTAM_{kvm}^{*\xi}$ is determined using constraints (1.14). These constraints ensure that the airline builds its calculation based on the actual departure time of aircraft $RTAM_{kvm}^{\xi}$that is determined by maintenance company through solving MSP. In this model, constraints (1.10) and (1.14) constitute the linkage between SOAMRPFD and MSP, as both problems are optimized conjunctionally.

A close look at the scheduled arrival and departure times of the aircraft in constraints (1.10) and (1.13), we can see that these times are function of accumulated propagated delay PD_{ikv}^{ξ}, which is due to the non-propagated delays caused by all previous covered flight legs. Based on this observation, it is obvious that the non-propagated delay is considered indirectly while calculating the scheduled arrival and departure times of the aircraft. It should be noted that in real practice, the scheduled arrival and departure times of the aircraft at the maintenance stations are two important factors that affect the aircraft maintenance sequencing (Beliën et al., 2013). Therefore, considering the non-propagated delay indirectly while determining the scheduled arrival and departure times of aircraft at maintenance stations, results in an indirect linkage between the non-propagated delay and the aircraft maintenance sequencing.

Forcing an aircraft that needs maintenance to undertake such an operation cannot be fulfilled through the coverage and balance constraints. Therefore, the operational restrictive constraints (1.15), (1.16), (1.17) and (1.18) are cast. Constraints (1.15) ensure that the maximum number of allowable take-offs is not violated by any aircraft in the fleet. Similarly, the accumulated flying times are restricted by constraints (1.16) and (1.17). In order to guarantee that the number of maintenance processes received by each aircraft is larger than or equal to one, constraints (1.18) are formulated. Given that our study has a planning horizon of 4-day and constraints (1.18) guarantee a greater than one maintenance process received by each aircraft, the constraint of one maintenance process every four days is accomplished. Finally, the constraints (1.19) - (1.23) represent the domain restrictions imposed upon the decision variables.

By looking at the objective function in Eq. (1.0), it is noticed that it is only dependent on the propagated delay happen between the flight legs, as shown by $\sum_{v=1,\dots,\Omega} C_{pD} \left(\sum_{k\in K} \sum_{i\in I} \sum_{j\in I} PD_{ijkv}^{\xi} x_{ijkv}^{\xi} \right)$, and there is no contribution from the aircraft maintenance sequencing, in terms of scheduled departure time of aircraft from maintenance stations $RTAM_{kvm}^{*\xi}$. The main reason is as follows. In fact, the main idea of our model is to solve the SOAMRPFD jointly with MSP, meaning that the scheduled departure times of aircraft $RTAM_{kvm}^{*\xi}$are determined based on the real departure times $RTAM_{kvm}^{\xi}$ calculated by maintenance stations. By doing so, we can avoid the delay occurs when covering the first flight leg after completing the maintenance. This means that the propagated delay will start when covering the later flight legs, which are included in the current terms of objective function. Based on the previous observation, the aircraft maintenance sequencing, known as the scheduled departure times, does not affect the overall delay. Therefore, the objective function is free from any factors regarding the aircraft maintenance sequencing.

The current formulation of the SOAMRPFD, including its objective function in Eq. (1.0) and its constraints in Eqs. (1.1)- (1.23) means that this model is solved for each disruption scenario. This point is apparent in two different parts. Firstly, it is apparent in the objective function, as it includes $\sum_{\xi\in\Psi} p^{\xi}$, which means that each disruption scenario is considered with its related occurrence probability. Secondly, it is also apparent in the constraints, as each constraint is formulated while considering each disruption scenario by inserting $\forall \xi \in \Psi$. It is noteworthy that the current formulation of SOAMRPFD is inspired by the formulation

presented in the work by Chen et al. (2002) who proposed a scenario-based stochastic formulation for capacity planning, in which different scenarios for the demand realizations are generated.

After presenting the SOAMRPFD and the procedures of generating the disruption scenarios, it is obvious that these scenarios are generated based on a time of one-year delay data, while respecting the overall percentage of delayed flights, the overall percentage of each cause of non-propagated delay, and the historical status of each flight leg. This helps in generating scenarios that can capture the disruption events happened during that year, resulting in a realistic representation for these events. Therefore, by doing so, there is no need to select these scenarios over time to ensure a realistic representation for them.

5. Lower-level optimization model (MSP decision model)

Referring to the description in Section 3.2, MSP, whose objective is to minimize the labor cost, is used to determine the workforce team sizes required to perform the maintenance processes to the aircraft. The MSP presented in this section is formulated based on a layered graph, which is commonly used in the literature when handling the staffing problem and the worker allocation problem (Yin & Wang, 2006). As shown in Fig. 5, the layered graph consists of number of layers, nodes, and arcs. Starting with the layers, each layer represents an aircraft that will receive the maintenance process, except the first and last layers which represent the source and sink nodes, respectively. Since MSP aims to determine the required workers for each aircraft represented as a layer, it is necessary to represent the workers in each layer. For this purpose, we insert nodes, which are the main components in each layer, so that the allowable number of workers available for performing maintenance processes can be represented. Finally, the arcs are incorporated in the graph in order to help in solving the model using ACO, especially when updating the pheromone trail.

Insert Fig. 5 appropriate here

Since SOAMRPFD, as a leader, is solved for different generated disruption scenarios, MSP as a follower should take a decision corresponding to each disruption scenario. Therefore, MSP is solved in response to each scenario, by using the decision variables, including wf^{ξ}_{kvsm} and $RTAM^{\xi}_{kvm}$, which determine the number of workers assigned for each aircraft, and the actual departure time of the aircraft from the maintenance station, respectively.

Before presenting the MSP decision model, we define its scope as follows:

- The workforce capacity of each maintenance station is deterministic.
- The decision model is formulated based on the existing shifts without regards to future changes.

Based on the predefined scope and notations, the optimization model for MSP, as a follower, can be written as below.

$$\min \sum_{\xi\in\Psi} p^{\xi} \left\{ \sum_{m\in MT} \sum_{s\in S} \sum_{kv\in K\Omega} C_{wkvsm}\, wf^{\xi}_{kvsm} \right\} \tag{2.0}$$

s.t.

$$RTAM^{\xi}_{kvm} \geq SDT^{\xi}_{kvm} + \left(SAT^{\xi}_{kvm} + TRT + \frac{l_{kv}}{wf^{\xi}_{kvsm}} - SDT^{\xi}_{kvm} \right)^{+} \qquad \forall\, kv \in K\Omega, m \in MT, \xi \in \Psi \tag{2.1}$$

$$\sum_{kv\in K\Omega} wf^{\xi}_{kvsm} \leq Q^{max}_{s} \qquad \forall\, s \in S, m \in MT, \xi \in \Psi \tag{2.2}$$

$$SAT_{kvm}^{\xi} = ATBM_{kvm}^{\xi} \quad \forall kv \in k\Omega, m \in MT, \xi \in \Psi \tag{2.3}$$

$$SDT_{kvm}^{\xi} = RTAM_{kvm}^{*\xi} \quad \forall kv \in k\Omega, m \in MT, \xi \in \Psi \tag{2.4}$$

$$wf_{kvsm}^{\xi} \in \{w_{sm}^{l}, \dots, w_{sm}^{u}\} \quad \forall kv \in k\Omega, s \in S, m \in MT, \xi \in \Psi \tag{2.5}$$

$$RTAM_{kvm}^{\xi} > 0 \quad \forall kv \in K\Omega, m \in MT, \xi \in \Psi \tag{2.6}$$

The objective function (2.0) serves to minimize the total labor cost. Calculation of the actual departure time for the aircraft from the maintenance station is represented by constraints (2.1). Here, in constraints (2.1), $\left(SAT_{kvm}^{\xi} + TRT + \frac{l_{kv}}{wf_{kvsm}^{\xi}} - SDT_{kvm}^{\xi}\right)^{+} = \max\left\{(SAT_{kvm}^{\xi} + TRT + \frac{l_{kv}}{wf_{kvsm}^{\xi}} - SDT_{kvm}^{\xi}), 0\right\}$. By looking precisely at constraints (2.1), we can notice that team size wf_{kvsm}^{ξ} can only affect the maintenance duration, represented by l_{kv}/wf_{kvsm}^{ξ}. As much the team size is large, the maintenance duration will be short. One of the question that might be asked is "where is the relation between the team size and turn-around times?". To answer this question, it is worth to remind that the turn-around times are the times consumed for getting passengers off, unloading the luggage, changing the gate, boarding, loading the luggage, moving the aircraft between the gate and maintenance station, and fueling the aircraft. All these activities during the turn-around time, are called the ground handling operations and are provided either by the airline itself or other service companies. This means that the maintenance companies do not participate on these operations. Therefore, there is no relationship between the team size assigned by maintenance companies and turn-around times.

In order to properly allocate the workers in carrying out the maintenance process, the worker capacity in each shift should be respected. Therefore, constraints (2.2) are formulated to guarantee that the designated number of workers for maintenance processes in each shift does not exceed each shift worker's capacity.

Acting as a follower to build an efficient staffing plan requires MSP to receive some information from the airline. For this purpose, linkage constraints (2.3) are incorporated in the model to represent the scheduled arrival time of each aircraft, whereas other linkage constraints (2.4) are cast to capture the scheduled departure time of the same aircraft. It should be noted that these linkage constraints are designed to ensure that the scheduled arrival and departure times, known as $SAT_{kvm}^{\xi}$and SDT_{kvm}^{ξ}, are calculated based on the decision variables ($ATBM_{kvm}^{\xi}$and $RTAM_{kvm}^{*\xi}$) received from the airline, as MSP is formulated as a follower and optimized in conjunction with SOAMRPFD. Finally, the domain of the decision variables is captured in constraints (2.5) and (2.6).

A close look at the formulation of the MSP, we can notice that it is solved in response to each disruption scenario received from the airline. This issue is obvious in two different parts of the model; the objective function and the constraints. For the objective function, it is formulated by inserting $\sum_{\xi \in \Psi} p^{\xi}$, which means that MSP will take decisions for each disruption scenario received from airline. For the constraints, on the other hand, they are formulated while considering each disruption scenario, as they include $\forall \xi \in \Psi$. From the previous description, the scenario linkage between SOAMRPFD and MSP becomes apparent.

6. Joint configuration optimization

In this section, we present a joint optimization model for the coordination between SOAMRPFD and MSP. This joint optimization is formulated as a Stackelberg game bi-level optimization model, in which the upper-level optimization model is represented by the SOAMRPFD that acts as a leader, whereas the lower-level optimization model is denoted by the MSP that acts as a follower. The decision variables (strategies) and objective functions (payoffs) are different for each model. Initially, SOAMRPFD, as a leader, determines its strategy which is sent to MSP. Next, MSP responds to the SOAMRPFD's strategy by choosing a strategy that maximizes its own payoff. The following equations summarize the bi-level optimization model for the coordination between SOAMRPFD and MSP:

$$min \sum_{\xi \in \Psi} p^{\xi} \left\{ \sum_{v=1,\ldots,\Omega} C_{pD} \left(\sum_{k \in K} \sum_{i \in I} \sum_{j \in I} PD_{ijkv}^{\xi} \, x_{ijkv}^{\xi} \right) \right\} \tag{3.0}$$

s.t. Constraints Eqs. (1.1) - (1.23) (3.1)

where given decision variables ($ATBM_{kvm}^{\xi}$and $RTAM_{kvm}^{*\xi}$) are used for solving:

$$\min \sum_{\xi \in \Psi} p^{\xi} \left\{ \sum_{m \in MT} \sum_{s \in S} \sum_{kv \in K\Omega} C_{wkvsm} \, wf_{kvsm}^{\xi} \right\} \tag{4.0}$$

s.t. Constraints Eqs. (2.1) - (2.6) (4.1)

From the above formulation, it is observed that SOAMRPFD makes decisions for the routing plan, including the maintenance visits through the decision variables $ATBM_{kvm}^{\xi}$ and $RTAM_{kv}^{*\xi}$. With these results as an input, MSP determine its staffing plan to satisfy the required maintenance processes through the decision variables wf_{kvsm}^{ξ} and $RTAM_{kvm}^{\xi}$. Next, MSP sends these decisions to SOAMRPFD. If these decisions influence the initial routing plan, SOAMRPFD further adjusts its decisions by resolving the model. This process is iterated until reaching the Stackelberg equilibrium, at which neither of the players are unwilling to adjust their decisions, because any change result in losses in their objective functions. Therefore, an optimal setting for the coordinated configuration is derived.

7. Solution method

Before describing our solution method, we present briefly the solution methods of existing bi-level optimization models. In the literature, there are two main solution methodologies; indirect and direct. Using the indirect methodology, the original bi-level model is converted to a single model, which is then solved by adoption of methods such as the B&B method based on Karushe-Kuhne-Tucker (KTT) conditions (Bard & Falk, 1982) and the penalty function method (Ren & Wang, 2016). Applying such indirect methodology to our model raises three issues. Firstly, it ignores the fact that each of the bi-level optimization problems are related to different companies with self-interest goals. Secondly, it ensures cooperation between the two companies that is not guaranteed in reality. Thirdly, it may lead to dominance of the follower's decision over the leader's decision, which will result in an incorrect representation for the proposed model. Therefore, in the light of these issues, using the indirect methodology to handle our bi-level optimization model is not a sound idea. The direct methodology, on the other hand, seeks the solution of the bi-level model directly by the adoption of specific methods, such as satisfactory solution methods (Muñoz & Abdelaziz, 2012). This type of methodology has certain advantages, such as maintaining the leader-follower structure and respecting the self-interest goals. However, using direct methodology becomes quite difficult in handling large-scale network optimization problems. This point is confirmed through the work by Wang

et al. (2016), in which it was mentioned that the direct method, such as satisfactory solution methods proposed by Muñoz & Abdelaziz (2012), could not solve a case study for power transformer product family and supply chain network. In particular, that case study consists of 10 power transformer base modules, 32 suppliers, 6 manufacturers, 4 assembly plants, and 3 distribution centers. This inability in solving this case study is due to the increase in network nodes. It is worth mentioning that the case study presented in our paper is even larger in size (i.e. number of nodes) compared to the case study by Wang et al. (2016). Indeed, our case study consist of a bi-level optimization model, in which the upper-level optimization model consists of 240 flight legs and 4 maintenance stations, whereas the lower-level optimization model consists of about 60 maintenance operations for aircraft and 360 potential team sizes. Since each upper-level and lower-level optimization model is related to a large-size network optimization problem, therefore, applying direct methodology such as satisfactory solution method to handle our bi-level optimization model is challenging.

As described in Sections (4) and (5), the upper-level optimization problem (SOAMRPFD) and the lower-level optimization problem (MSP) are essentially modeled as network-based problems, which can be efficiently solved with ACO due to its ability to handle large and complex network-based problems (Wu et al., 2009; Deng & Lin, 2011; Yu & Yang, 2011; Balseiro et al., 2011). All these observations motivate us to develop a bi-level nested ACO approach, to address the interdependence between each level of the problems, whilst each individual level is solved by one specific ACO. The nested ACO can capture the existing link between the leader and the follower by way of feeding back the decision variables. Due to the fact that each level of the bi-level model has distinctive features and goals, we explain ACO for solving each level. Next, the nested ACO algorithm is explained.

7.1 ACO for the upper-level optimization

In this section, the main two parts of the ACO adopted to solve SOAMRPFD are as follows:

1) Route construction. Since SOAMRPFD is represented by a modified connection network, as shown in Fig. 4, it is appropriate for ACO to construct the required routes. This part is conducted with the help of an ant (i.e. each ant simulates an aircraft $k \in K$) that moves throughout the network. Each ant constructs its route by building a path, starting at the source node and ending at the sink node. Between the source and sink nodes, the ant scouts for appropriate flight nodes to be added to its route by the usage of the state transition rule. To clarify this point, assume an ant is currently positioned at flight leg node i and it intends to select its next flight leg node j. This selection is done based on the following state transition rule:

$$j = \begin{cases} \arg\_max_{j \in N_i^k} \left\{ \left[\tau_{ij}{}^{\xi}\right]^{\alpha} \left[\eta_{ij}{}^{\xi}\right]^{\beta} \right\} & if\ q \le q_0 \\ J & if\ q > q_0 \end{cases} \qquad (5.1)$$

where N_i^k represents the group of possible flight legs that ant k can select after covering flight leg i. The term $\tau_{ij}{}^{\xi}$ is the pheromone trail of the ordinary arc $ord(i,j)$ that results during consideration of the disruption scenario ξ. On the other hand, $\eta_{ij}{}^{\xi}$ is the heuristic function of the same arc, which is equal to $1/(C_{pD} * PD_{ijkv}^{\xi})$. The relative importance of $\tau_{ij}{}^{\xi}$ and $\eta_{ij}{}^{\xi}$ is denoted by parameters α and β, respectively. q_0 is the exploration threshold parameter ($0 \le q_0 \le 1$) and q represents random number that is generated according to the uniform distribution [0~1]. Ideally, the value q determines the selection of the next flight leg j. If $q \le q_0$, flight leg j is selected because its arc

$ord(i,j)$ carries the greatest ${\tau_{ij}}^{\xi}$ and ${\eta_{ij}}^{\xi}$. Conversely, if $q > q_0$, flight leg j is selected using the probability rule below:

$$P_{ij}^{k} = \frac{\left[{\tau_{ij}}^{\xi}\right]^{\alpha}\left[{\eta_{ij}}^{\xi}\right]^{\beta}}{\sum_{j \in N_i^k}\left[{\tau_{ij}}^{\xi}\right]^{\alpha}\left[{\eta_{ij}}^{\xi}\right]^{\beta}} \qquad if\ j \in N_i^k \qquad (5.2)$$

This selection part continues until all the flight legs are covered by the aircraft. It should be pointed out that when the upper-level ACO is adopted to resolve SOAMRPFD in subsequent iterations, a new routing plan is built. This includes changing in the leg-to-leg connections that are predetermined in the first iteration, as the upper-level ACO seeks to minimize the expected cost of propagated delay. The changes in leg-to-leg connections occur in two different situations; connecting between maintenance stations and flight legs, and connecting between flight legs.

To clarify the first situation, suppose that, in the first iteration of the upper-level ACO, the airline gets a solution for SOAMRPFD as shown in Fig 1. By looking at the solution, we can see that the first aircraft will cover flight 5 then visit the maintenance station. On completion of the maintenance, the aircraft will resume covering flight legs by flying flight 8. Imagine, for instance, due to manpower restrictions, the maintenance station cannot complete the maintenance as planned by airline, meaning different departure time for the aircraft. Accordingly, covering flight 8 will be delayed. In this case, when the airline uses the upper-level ACO to resolves the SOAMRPFD in subsequent iterations, the different departure time for aircraft determined by the current maintenance station will be considered. In this connection, the upper-level ACO will try to find out another flight instead of flight 8, in order to minimize the expected cost of propagated delay. If flight 8 is replaced by another flight, it is highly probable to change all its subsequent leg-to-leg connections.

To clarify the second situation, which is connecting between flight legs, we use the same solution shown in Fig 1. By looking at the solution, it is noticed that the first aircraft starts its route by covering flight 1, then covering flights 3 and 4. Suppose that the expected cost of the propagated delay caused by connecting flights 1 and 3 is 125. Imagine, for instance, when the upper-level ACO is adopted to resolve SOAMRPFD in subsequent iterations, it finds another connection for flight 1, say connection between flights 1 and 23, such that the expected propagated delay cost for this connection is 75. As a result, in subsequent iteration, the ACO will change the connection between flights 1 and 3, to be between flights 1 and 23. Since flight 3 is replaced by flight 23, it is highly probable to change all the subsequent leg-to-leg connections. From these two situations, it is obvious that, when resolving the SOAMRPFD, the predetermined leg-to-leg connections are not fixed.

2) Updating the pheromone trail: This part considers the core part of ACO, where the pheromone trail is updated to reflect the quality of the current solution. Taking ρ as the evaporation rate parameter $(0 < \rho < 1)$, the pheromone trail can be updated in accordance with the following rule:

$$\tau_{ij,new}^{\xi} = (1-\rho)\tau_{ij,old}^{\xi} + \Delta\,{\tau_{ij}}^{\xi} \qquad (5.3)$$

For each iteration, the first term $(1-\rho)\tau_{ij,old}^{\xi}$ is utilized to achieve a uniform reduction of the pheromones. The advantage is that in the next iterations, the ants are able to ignore the bad routes and search for better ones. In a disruption scenario ξ, the pheromone quantity on the edge $ord(i,j)$ is defined by the second term $\Delta\,{\tau_{ij}}^{\xi}$, which is only applied to update the phermone trails of the edges that constitute the best solution so far. Thus, such updating enables the ants to look for better

routes in the subsequent iterations. By taking Q as the control factor in depositing the pheromone, then $\Delta\,\tau_{ij}{}^{\xi}$ is calculated using Eq. (5.4)

$$\Delta\,\tau_{ij}{}^{\xi} = \frac{Q}{cost\left(A_{best}^{\xi}\right)} \qquad if\{i,j\} \subseteq A_{best}^{\xi} \qquad (5.4)$$

The value of Q specifies either a local optimal convergence or a random search for the algorithm. The $cost\left(A_{best}^{\xi}\right)$ represents the lowest propagated delay cost between the start and the present iteration, and ξ is the handling disruption scenario.

7.2 ACO for the lower-level optimization

Similar to Section 7.1, the main two parts of ACO used to solve MSP are described as follows:

1) Staffing construction: Thanks to the layered graph representation for MSP, it eases the process of building the staffing plan for each maintenance station. The staffing plan is built by the ants by starting from the source node, visiting each layer (i.e. each layer represents an aircraft that will receive the maintenance process) in sequence by selecting a node with an appropriate team size, and eventually ending at the sink node. To clarify the step of selecting the appropriate team size, suppose an ant currently visits layer bv and the next layer kv contains the team size to be selected. This selection is applied according to the following state transition rule:

$$w = \begin{cases} \arg\_max_{w \in N_{kv}^{\prime ant}} \left\{ \left[\tau_{bvkvw}^{\xi}\right]^{\alpha'} \left[\eta_{bvkvw,worker}^{\xi}\right]^{\beta'} \right\} & if\ q \le q_0' \\ W & if\ q > q_0' \end{cases} \qquad (6.1)$$

where $N_{kv}^{\prime ant}$ is the set of potential team sizes that can be determined to maintain the aircraft in layer kv. The term τ_{bvkvw}^{ξ} is the pheromone trail of the edge connecting layers bv and kv, such that layer kv will be served by team size w, under disruption scenario ξ. On the other hand, $\eta_{bvkvw,worker}^{\xi}$ is the heuristic function for the same edge, which is equal to $1/wf_{kvsm}^{\xi}$. The two parameters $\alpha'$and β'represent the relative importance of τ_{bvkvw}^{ξ} and $\eta_{bvkvw,worker}^{\xi}$, respectively. q_0' is the exploration threshold parameter $(0 \le q_0' \le 1)$ and q represents random number that is generated according to the uniform distribution [0∼1]. Ideally, value q determines a team size to be selected by the ant. If $q \le q_0'$, then it selects a team size w in which its edge has the best information heuristic and pheromone trail. Conversely, if $q > q_0'$, the ant chooses a team size w according to the following probability rule:

$$P_{bvkvw}^{ant} = \frac{\left[\tau_{bvkvw}^{\xi}\right]^{\alpha'} \left[\eta_{bvkvw,worker}^{\xi}\right]^{\beta'}}{\sum_{l \in N_{kv}^{\prime ant}} \left[\tau_{bvkvw}^{\xi}\right]^{\alpha'} \left[\eta_{bvkvw,worker}^{\xi}\right]^{\beta'}} \qquad if\ w \in N_{kv}^{\prime ant} \qquad (6.2)$$

2) Updating the pheromone trail: This process is done using Eq. (6.3)

$$\tau_{bvkvw,new}^{\xi} = (1-\rho')\tau_{bvkvw,old}^{\xi} + \Delta\,\tau_{bvkvw}{}^{\xi} \qquad (6.3)$$

Similar to the ACO used for OSAMRPFD, the first term $(1-\rho')\tau_{bvkvw,old}^{\xi}$ is applied after each ant completes its path, whereas the second term $\Delta\,\tau_{bvkvw}{}^{\xi}$ is utilized to update the the phermone

trails of the edges that form the best solution so far. $\Delta\,\tau_{bvkvw}{}^{\xi}$ represents the pheromone quantity on the edge connected between the layers bv and kv, such that the aircraft in layer kv is served by team size of w, under disruption scenario ξ. To calculate $\Delta\,\tau_{ij}{}^{\xi}$, we use the following rule:

$$\Delta\,\tau_{bvkvw}{}^{\xi} = \frac{Q'}{cost\left(B^{\xi}_{wm,best}\right)} \qquad if\ edge \subseteq B^{\xi}_{wm,best} \qquad (6.4)$$

where Q' is the factor controlling the process of pheromone depositing. The $cost\left(B^{\xi}_{wm,best}\right)$ denotes the lowest cost achieved from the start to the present iteration, while determining the staffing plan for maintenance station m, under disruption scenario ξ.

7.3 Bi-level nested ACO algorithm for leader-follower optimization

The solution of joint optimization of SOAMRPFD and MSP, including interdependence, necessitates an algorithm that captures this issue. For this purpose, a bi-level nested ACO algorithm is proposed. It starts with the upper-level ACO in order to solve SOAMRPFD and makes decisions regarding the routing plan and maintenance visits using variables $ATBM^{*\xi}_{kvm}$ and $RTAM^{*\xi}_{kvm}$. The upper-level solution is sent to the lower-level ACO as an input in order to solve MSP and make decisions regarding the staffing plan using decision variables wf^{ξ}_{kvsm} and $RTAM^{\xi}_{kvm}$ to satisfy the received solution. All the lower-level best solutions are relayed to the upper-level ACO to be re-run and its solution adjusted. The algorithm undergoes some iterations until it arrives at the Stackelberg game equilibrium. The following is a detailed stepwise procedure for implementing the bi-level nested ACO algorithm:

Step 1.0: Set the initial value for the upper-level and the lower-level ACOs' parameters including; ($\alpha, \beta, q_0, \rho, Q, \alpha', \beta', q_0', \rho', Q'$, number of ants for each ACO).

Step 1.1: Create a specific number of disruption scenarios. Then, store these scenarios in a list named (Ψ).

Step 1.2: Calculate the value of (Ω), which indicates the maximum average number of maintenance processes that each aircraft should receive. Next, set the value for the maximum number of iterations.

Step 1.3: Set the number of iterations to be one.

Step 1.4: Apply the upper-level ACO by carrying out the following Steps 1.5 - 1.13:

Step 1.5: Examine the condition of the Ψ list. Proceed to Step 1.6 if it is not empty, else move to step 1.12.

Step 1.6: Select a single disruption scenario ξ from the Ψ list.

Step 1.7: Put all the aircraft operated by the airline in a list called (K). Note that each aircraft is denoted by an ant. Store all the flight leg nodes that should be covered by the aircraft in another list named (I).

Step 1.8: To initiate the construction of the routes, go through the next sub-steps:

- Step a: Using the K list, examine its condition. Proceed to Step 1.9 if it is empty, else select an ant or aircraft k and place it on the source node of the network shown in in Fig. 4.
- Step b: Examine the condition of the I list with respect to the coverage constraints in Eq. (1.2). Proceed to Step c if it is not empty, else move to step 1.9.
- Step c: According to the constraints defined in Eq. (1.3), let ant k starts its routes by covering a flight leg i from the I list.
- Step d: By scanning through the I list, select the possible flight legs to be covered by taking the constraints defined in Eqs. (1.7) and (1.8) into consideration. Proceed to Step j if there are no more possible flights to be covered, else move to Step e.

Step e: Using the state transition equation denoted in Eqs. (5.1) and (5.2), choose the flight leg j among the possible flight legs to be covered.

Step f: Using the constraints given in Eqs. (1.15) - (1.18), determine if there is a violation of the operational maintenance constraints after selecting flight leg j. Proceed to Step g if the constraints are violated, else move to Step i.

Step g: Using the constraints shown in Eqs. (1.9) and (1.10) as a basis, draw up a maintenance visit.

Step h: On completing the maintenance, proceed to cover the flight legs by taking these constraints (Eqs. 1.5, 1.6, and 1.11 - 1.14) into consideration. Note that the constraints in Eq. (1.13) are considered only when the number of iterations=1, as discussed in Section 4. For the remaining iterations, we consider the constraints in Eq. (1.14), which contains the decisions ($RTAM_{kvm}^{\xi}$) stored in Step 2.8.

Step i: Using the I list, exclude the selected flight leg j and insert it to the constructed route. Next, repeat the process of selecting another flight leg by moving to step d.

Step j: Place aircraft k in the sink node and terminate its route, while considering the constraints shown in Eq. (1.4).

Step k: Using the K list, exclude aircraft k and proceed to step a.

Step 1.9: For all arcs of the obtained solution, keep the pheromone trails up-to-date by using Eqs. (5.3) and (5.4).

Step 1.10: For the existing iteration, calculate the solution regarding the disruption scenario ξ (Z_{iter}^{ξ}). If a better solution is obtained, revise the best solution for the disruption scenario ξ (Z_{best}^{ξ}).

Step 1.11: Save the best solution of the present disruption scenario Z_{best}^{ξ} and the decision variables ($ATBM_{kvm}^{\xi}$ and $RTAM_{kvm}^{*\xi}$) that are related to it. Eliminate ξ from the Ψ list and move on to Step 1.5.

Step 1.12: Augment the best solution gotten from each disruption scenario ($Z_{iter,routing} = \sum_{\xi \in \Psi} Z_{best}^{\xi} * p^{\xi}$) to examine the solution of the present iteration.

Step 1.13: Check whether the stopping criteria for the upper-level ACO (SAC-UL) is satisfied or not, given the satisfaction status of the lower-level ACO stopping criteria (SAC-LL). There are three possibilities for this step, as follows:

- If SAC-UL is satisfied or not, while SAC-LL is not satisfied, then go to Step 1.14.
- If SAC-UL is not satisfied, while SAC-LL is satisfied, then use the same list generated in Step 1.1 to update the Ψ list, increase the number of iterations, and proceed to Step 1.4.
- If both SAC-UL and SAC-LL are satisfied, then go to Step 2.11.

Step 1.14: Apply the lower-level ACO by carrying out Steps (2.1) - (2.10).

Step 2.1: Prepare a routing list (RL^{ξ}) that stores the routing solution under different disruption scenarios received from Step 1.11. For each routing solution, prepare a maintenance station list (MT^{ξ}) that contains the maintenance stations that will be visited by the aircraft.

Step 2.2: Examine the condition of the RL_m^{ξ} list. Proceed to Step 2.3 if it is not empty, else move to Step 2.9.

Step 2.3: Select a routing solution rl_m^{ξ} from the RL_m^{ξ} list.

Step 2.4: Examine the condition of the MT^{ξ} list. Proceed to Step 2.5 if it is not empty, else move to Step 2.7.

Step 2.5: Select a maintenance station m^{ξ} from the MT^{ξ} list. Based on the routing solution rl_m^{ξ}, extract all the aircraft that will be maintained at the maintenance station m^{ξ}, and store them in a list called ($K\Omega$).

Step 2.6: Build the staffing plan by executing these sub-steps:

Step a: Construct the ANT list.

Step b: Pick an ant from the ANT list and put it in the source node of the graph represented in Fig. 5. Mark this position as an initial position for the ant in its constructed path.

Step c: Check the status of the $K\Omega$ list as to whether all the aircraft are visited by the ant or not. If all aircraft are visited, then proceed to Step i, else move to Step d.

Step d: Pick the first unvisited kv aircraft from the $K\Omega$ list. According to the graph in Fig. 5, mark the picked aircraft's layer as a next position to be visited by the ant.

Step e: Calculate the scheduled arrival and departure time for the picked aircraft, while considering the constraints mentioned in Eqs. (2.3) and (2.4), which contains decisions received from the upper-level ACO.

Step f: Based on results of Step e, determine the targeted shift. Then, determine the potential sizes for the team that can be built, while considering the constraints stated in Eqs. (2.2) and (2.5).

Step g: Select the team size according to the state transition equation described in Eqs. (6.1) and (6.2).

Step h: Put the ant on the aircraft in kv layer represented by the graph in Fig. 5, mark this position as an initial position for the next selection. Then, mark the aircraft kv as a visited aircraft on the $K\Omega$ list and go to Step c.

Step i: Put the ant in the sink node and terminate its path.

Step j: Use the rule represented in Eqs. (6.3) and (6.4) to keep the pheromone trails up-to-date.

Step k: Examine the condition of the ANT list. Proceed to Step l if it is not empty, else move to step m.

Step l: Update all the aircraft in $K\Omega$ list to be unvisited aircraft and proceed to Step c.

Step m: Assess the solution of the present iteration (Z_m^{ξ}), update the best solution ($Z_{m,best}^{\xi}$) if needed, and proceed to Step 2.4.

Step 2.7: Compute the best solution for the staffing plan under disruption scenario ξ, calculated as $Z_{total,m,best}^{\xi} = \sum_{m \in MT} Z_{m,best}^{\xi}$. Then, proceed to Step 2.2.

Step 2.8: Store the best solution of each disruption scenario ξ, and its related decision variables ($RTAM_{kvm}^{\xi}$).

Step 2.9: Assess the solution of the present iteration $Z_{iter,staffing} = \sum_{\xi \in \Psi} Z_{total,m,best}^{\xi} * p^{\xi}$.

Step 2.10: Check whether the stopping criteria for the lower-level ACO (SAC-LL) is satisfied or not, given the satisfaction status of the upper-level ACO stopping criteria (SAC-UL). There are three possibilities for this step, as follows:

- If SAC-LL is satisfied or not, while SAC-UL is not satisfied, then use the same list generated in Step 1.1 to update the Ψ list, increase the number of iterations, and proceed to Step 1.4.

- If SAC-LL is not satisfied while SAC-UL is satisfied, then increase the number of iterations, and proceed to Step 1.14.
- If both SAC-LL and SAC-UL are satisfied, then proceed to Step 2.11.

Step 2.11: Return the best solution generated by both levels. Since these solutions generate the status in which both players are unwilling to adjust their decisions, the Stackelberg equilibrium is derived, and we terminate the algorithm.

Fig. 6 presents the flow chart of the bi-level nested ACO algorithm. The left-hand side of the figure shows the upper-level ACO used to solve the SOAMRPFD, whereas the lower-level ACO that is used to solve the MSP is shown on the right-hand side of the figure. For computational efficiency and a reasonable problem context, we set the stopping criteria for the bi-level ACO to be the convergence (i.e. 100 successive iterations without solution improvement), or when completing the maximum number of iterations (i.e. 500 iterations), whichever comes first. If both stopping criteria for the bi-level are satisfied, then the nested algorithm is terminated, and the Stackelberg equilibrium is derived.

From the description of the bi-level nested ACO, we can see that the existence of the Stackelberg equilibrium is mainly dependent on two main points: (1) finding feasible solutions to SOAMRPFD and MSP, then (2) reaching the convergence points using the obtained feasible solutions.

Regarding the feasible solutions, imagine, if, for instance, the maximum number of iterations is completed and the feasible solution of one model, SOAMRPFD or MSP, cannot generate a feasible one for the other model. This situation means that there is no feasible staffing plan by maintenance companies to meet the requirement of the routing plan by airline or vice versa. Accordingly, the convergence points cannot be reached and the Stackelberg equilibrium cannot be derived. In this situation, we need to rerun the bi-level nested ACO algorithm, but two questions might be asked here. Firstly, should the reruns experiments be conducted using the same conditions as in the first run, or different conditions. If the answer is, reruns with different conditions, what are the factors that should be changed. Secondly, how many runs are required to find out feasible solutions. For the first question, the conditions of reruns, in fact, if the reruns experiments are conducted using the same conditions, it is highly probable to face a deadlock situation, where there are no feasible solutions, so that the convergence points cannot be reached and the Stackelberg equilibrium cannot be derived. This situation happened once during our preliminary experiments, when setting unreasonable value for the assumed maintenance duration by airline MAT=5. Indeed, this maintenance duration is short and quite difficult to be realized, as it requires large team sizes from maintenance companies, which sometimes go beyond the workforce capacity. This finally results in infeasible solutions even when rerunning the algorithm many times using the same conditions. Therefore, in the light of this observation, the reruns experiments should be conducted using different conditions. In our preliminary experiments, it was noticed that the most critical factor that influences the feasibility of the solutions, is the assumed maintenance duration MAT. This factor is assumed by the airline in the first iteration of the algorithm. If this duration is relatively short, less than 6 hours, it means large team sizes required from the maintenance companies, which sometimes difficult to be realized due to workforce limitations, so that feasible solutions cannot be achieved and then the Stackelberg equilibrium at the convergence points cannot be derived. On the other hand, if this duration is reasonable, from 6 to 9 hours, it will be easier for maintenance companies as they need to provide medium and small team sizes, resulting in finding out feasible solutions. Then, the convergence points can be reached and the Stackelberg equilibrium can be derived. Based on this observation, in the rerun experiments, we change the condition of the experiment by

extending the assumed maintenance duration MAT to be within 6 to 9 hours. Regarding the second question, the number of required reruns, it was recommended in the literature to set the maximum number of runs at 100 runs (Y. Yu & Huang, 2010). If the maximum number of runs is completed without finding out feasible solutions, which was very rare during our preliminary experiments, this means that convergence points cannot be reached and the Stackelberg equilibrium cannot be derived, resulting in unachievable cooperation among the players.

Moving to convergence after obtaining the feasible solutions, imagine a situation, in which the maximum number of iterations is completed, and feasible solutions are obtained, but the convergence points are not reached. This means that both players keep adjusting their discussions, resulting in underived Stackelberg equilibrium. In this situation, we also need to rerun the algorithm and the same questions about the conditions of reruns and the number of runs might be asked here. Regarding these questions, in our preliminary experiments, we find out that rerunning the experiments using the same conditions of the first run can easily help in reaching the convergence points in few number of runs. Therefore, based on this observation, to find out the convergence points, we need to rerun the bi-level nested ACO, using the same conditions of the first run and the same maximum number of runs that is mentioned earlier. If the maximum number of runs is completed without reaching the convergence points, which was also rare in our preliminary experiments, it means that the Stackelberg equilibrium cannot be derived and the cooperation among the players cannot be achieved.

Insert Fig. 6 appropriate here

7.4 Evaluating the algorithm performance

One of the obvious question after using the bi-level nested ACO algorithm is how close to optimality the obtained solutions are. In ideal situations, we would like to use the optimal solution as a benchmark, but sometimes it is challenging to get the optimal solution, as in our bi-level optimization model. This challenge is apparent in each level of the bi-level optimization model. For the upper-level optimization model, known as the SOAMRPFD, it was proved that it belongs to the class of NP-hard problem, even in its deterministic form (Eltoukhy et al., 2018). In our study, the upper-level optimization model is a stochastic routing model that considers large number of disruption scenarios. In addition, the upper-level optimization model is validated through a real case study that consists of large number of flight legs and aircraft. Moving to the lower-level optimization mode, known as the MSP, it is also NP-hard problem (Yin & Wang, 2006). Moreover, in our study, the MSP is solved in a response to each disruption scenario received from the SOAMRPFD. All the previous observations result in more complexity being added to the bi-level optimization model, therefore, it is challenging to find out the optimal solution. To compromise this situation, we propose finding out an approximated lower bound for each of SOAMRPFD and MSP. Then, calculating the gap between the obtained solution and the lower bound, to be the criterion for evaluating the algorithm performance.

To find out an approximated lower bound of the SOAMRPFD, we propose an algorithm, which its main idea plays around generating all the feasible routes and selecting the routes with the lowest expected propagated delay cost (Liang et al., 2015). The main procedure of the algorithm can be summarized as follows:

Step 1: Generate a number of disruption scenarios. Then, store them in a list called (Ψ).

Step 2: Construct two lists, such that the first one contains the aircraft (K), whereas the second one contains the flight legs (I).

Step 3: Check the status of Ψ list. If it is not empty, pick a disruption scenario ξ from the Ψ list, otherwise go to Step 10.

Step 4: Check the status of K list. In case of nonempty K list, pick aircraft k, otherwise proceed to Step 9.

Step 5: Using the I list, examine its condition. Proceed to Step 6 if it is not empty, else go to Step 9.

Step 6: Generate all the feasible routes for the picked aircraft k, while considering the constraints stated in Eqs. (1.2) – (1.23). In our preliminary experiments, we observed that the expected propagated delay cost increases significantly, when the maximum number of take-offs goes beyond 7 take-offs. This is because the accumulated propagated delay increases with the number of flights or take-offs in the route, resulting in an increase in the expected propagated delay cost. For example, when handling a case of 240 flight legs that are covered by 30 aircraft, the expected propagated delay cost was about 1604.85 and 2899.78 in case of 7 take-offs and 8 take-offs as the maximum, respectively. This observation reveals that going beyond 7 take-offs as the maximum results in a generation of unprofitable routes. Therefore, to generate profitable routes and save the long computational time consumed in generating all the feasible routes, we restrict this step to generate feasible routes with 7 take-offs as the maximum.

Step 7: Calculate the expected propagated delay cost for each generated route using Eq. (1.1). Next, pick the route with the lowest cost, and store its cost in Z_k^{ξ}.

Step 8: Assign the picked route to aircraft k. Next, exclude all the flight legs included in the picked route from the I list, remove aircraft k from the K list, and go to Step 4.

Step 9: Calculate the lower bound for disruption scenario ξ, as follows; $LB_{routing}^{\xi} = \sum_{k \in K} Z_k^{\xi}$. Then, go to Step 3.

Step 10: Calculate the overall lower bound for all disruption scenarios, as follows; $LB_{routing} = \sum_{\xi \in \Psi} LB_{routing}^{\xi} * p^{\xi}$. Then, terminate the algorithm.

Moving to the MSP, to find its approximated lower bound, we propose another algorithm for this purpose. By looking at the MSP, we can see that the team sizes are formed while respecting two limits; the minimal team sizes, and the maximal team sizes. This observation is exploited as the main idea of our algorithm, such that the teams with the possible minimal sizes are selected to perform the maintenance for the aircraft. Since the possible minimal sizes are selected, the lowest labor cost can be achieved. This finally results in finding out the approximated lower bound of the MSP. The main steps of this algorithm can be described as follows:

Step 1: Store the solution generated in the previous algorithm in a list, called the routing solution RL^{ξ}. It should be noted that the stored solutions in this step are the solutions achieved at the lower bound of SOAMRPFD. For the solution generated under a specific disruption scenario, construct a maintenance station list (MT^{ξ}) to store the maintenance stations that will be visited by the aircraft.

Step 2: Check the status of the RL_m^{ξ} list. If it is not empty, select a routing solution rl_m^{ξ} from the RL_m^{ξ} list, otherwise move to Step 10.

Step 3: Using the MT^{ξ} list, check its condition. If it is not empty, pick a maintenance station m^{ξ} from the MT^{ξ} list, else move to Step 9.

Step 4: Using the routing solution rl_m^{ξ}, extract all the aircraft that will receive the maintenance process at the maintenance station m^{ξ}, and store them in a list called ($K\Omega$).

Step 5: Examine the status of the $K\Omega$. In case of nonempty $K\Omega$, proceed to Step 6, otherwise move to Step 8.

Step 6: Pick aircraft kv from the $K\Omega$ list. Then, determine the possible team sizes that can be formed using constraints stated in Eqs. (2.2) and (2.5) and calculate their related labor cost.

Step 7: Select the team size with the lowest labor cost, which most likely is the team with the smallest possible size. Then, store the related cost of the selected team in $Z_{kv,m}^{\xi}$, and go to Step 5.

Step 8: Calculate the lower bound for maintenance station m, while handling disruption scenario ξ, as follows; $LB_{staffing,m}^{\xi} = \sum_{kv \in K\Omega} Z_{kv,m}^{\xi}$. Then, go to Step 3.

Step 9: Calculate the lower bound for all maintenance stations, while handling disruption scenario ξ, as follows; $LB_{staffing}^{\xi} = \sum_{m \in MT} LB_{staffing,m}^{\xi}$. Then, move to Step 2.

Step 10: Calculate the overall lower bound for all disruption scenarios, as follows; $LB_{staffing} = \sum_{\xi \in \Psi} LB_{staffing}^{\xi} * p^{\xi}$. Then, terminate the algorithm.

8. Case study

8.1 Data setting

After developing a method that can allow coordination among the decision makers of SOAMRPFD and MSP, it is necessary to demonstrate the validity of the method. To do so, a case study for a Middle Eastern major airline and a maintenance company is presented. As discussed earlier, SOAMRPFD is presented with the objective of minimizing the expected propagated delay cost. To assess the performance of SOAMRPFD in minimizing the propagated delay, the fleet with the largest minutes of delay is selected and its associated data are collected. These data are presented in the upper part of Table 3. In order to build a feasible routing plan for the airline, its aircraft need to visit the maintenance company to receive maintenance service. Hence, it becomes imperative to collect data for the visited maintenance stations to execute a real case study. The collected data from the maintenance company includes the workforce capacity, shifts, etc. In our case study, the maintenance company has four stations, and each station is located in different airports. From the acquired data, it was noted that these stations have more or less the same conditions. Therefore, for simplicity, we assume that all the stations have the same conditions in terms of number of shifts, team sizes, and costs, as shown in the lower part of Table 3.

Table 3: Detailed description of data acquired from airline and maintenance companies.

Airline	
I	240 flight legs
KT	30 aircraft
C_{max}	10 take-offs
T_{max}	40 hours
A	8 airports
MT	4 maintenance stations

TRT	45 minutes							
MAT	8 hours							
C_{pD}	$C_{pD} = \begin{cases} 75 & PD_{ijkv}^{\xi} \leq 15\ minutes \\ 125 & PD_{ijkv}^{\xi} > 16\ minutes \end{cases}$							
Maintenance company								
• Morning shift								
w_{Sm}^{l}	8 workers							
w_{sm}^{u}	15 workers							
Q_s^{max}	150 workers							
Potential team size (wf_{kvsm}^{ξ})	8	9	10	11	12	13	14	15
Cost of the team size (C_{wkvsm})	720	800	950	1000	1050	1120	1180	1250
• Afternoon shift								
w_{Sm}^{l}	5 workers							
w_{sm}^{u}	10 workers							
Q_s^{max}	100 workers							
Potential team size (wf_{kvsm}^{ξ})	5	6	7	8	9	10		
Cost of the team size (C_{wkvsm})	670	730	800	870	950	1020		
• Night shift								
w_{Sm}^{l}	2 workers							
w_{sm}^{u}	5 workers							
Q_s^{max}	50 workers							
Potential team size (wf_{kvsm}^{ξ})	2	3	4	5				
Cost of the team size (C_{wkvsm})	670	720	790	830				
l_{kv}	50 hours							

The experiments on this case study were performed on an 8 GB RAM Windows 10 laptop, equipped with an Intel i7 CPU with a clock speed of 2.50 GHz. MATLAB R2014a was utilized to code the proposed model and algorithm.

8.2 Scenario generation

It is worth mentioning that, in real practice, a larger number of disruption scenarios results in a better representation to the randomness of the non-propagated delay, but it requires a higher computational burden. Therefore, one of the obvious questions before implementing our case study is how many the generated disruption scenarios are enough to reflect real situations without requiring high computational burden. To answer this question, computational experiments were conducted, in which the SOAMRPFD was solved under a different number of equally likely scenarios, starting from 50 up to 150, as recommended by experts in the airline. For this purpose, we collect the flight information for all the case study flights recorded by the major Middle Eastern airline from January 2016 to December 2016. The collected information includes features such as flight number, departure airport, departure time, arrival airport, arrival time, flight duration, and the non-propagated delay for each flight. In each experiment, a number of scenarios was generated using the procedure explained in section 4, so that we have realistic

representation for the disruption events. Next, the SOAMRPFD with a 4-day planning horizon was solved and the value of objective function, expected propagated delay cost, was calculated and illustrated in Fig. 7. So, the planning horizon and the realistic representation of disruption events, known as the disruption scenarios, are considered while calculating the results presented in Fig. 7. The question of why Fig. 7 collapses sequence-dependent issue. Indeed, this issue is considered in the constraints (1.1) of the model. Therefore, this point is taken into consideration while solving the model, which its results are presented in Fig. (7).

We can see from this figure that, when the number of generated scenarios is low, say 50 scenarios, we obtain a low or underestimated value for the expected propagated delay cost. On the other hand, when the number of generated scenarios increases, we get better estimation for the expected propagated delay cost. The rationale behind the aforementioned situation lies in the fact that when the number of generated scenarios is low, the randomness of the non-propagated delay is not well represented. So, it is highly probable to overlook some situations for the delay, resulting in a low (underestimated) expected propagated delay cost. As the number of scenarios increases, we get better representation to the randomness of the non-propagated, so that overlooking some situations for the delay can be avoided. This finally results in a better estimation for the expected propagated delay cost. By looking at the results represented at Fig. 7, we can see that the appropriate number of scenarios needed to reflect the real situation is 100, as considering more than 100 scenarios does not lead to any significant improvement in the estimation of the expected propagated delay cost. This number of scenarios was confirmed by two parties. Firstly, when the experts of the airline were initially consulted, they recommended that 100 equally likely scenarios were more than enough to capture the real disruption situations. Secondly, from the literature, this number is confirmed through the work by Yen and Birge (2006), who generated disruption scenarios for flight delays, while solving the crew scheduling problem. Therefore, based on these observations, we decided to set the number of scenarios to be 100 throughout our case study.

Insert Fig. 7 appropriate here

8.3 Parameter setting of bi-level nested ACO algorithm

To adopt the bi-level nested ACO algorithm as a solution method, it is necessary to find out the best setting of the algorithm's parameter, as different parameters with different setting are expected to influence the performance of the ACO algorithm (Deng & Lin, 2011). Towards this goal, the most important parameters that influence the ACO performance should be selected, then different levels of the selected parameters should be determined. By doing so, pheromone trail importance, heuristic function importance, exploration threshold, evaporation rate, and ant size are selected as the most important parameters (Dorigo & Stützle, 2004). For the selected parameters, different levels are determined for the upper-level ACO and the lower-level ACO, as listed in Tables 4 and 5, respectively. It should be noted that these parameters' levels are determined as being commonly used in the literature (Deng & Lin, 2011; Yin & Wang, 2006).

Table 4: Parameters levels of the upper-level ACO.

Parameter	Level 1	Level 2	Level 3	Level 4
α	1	2	3	4
β	1	2	3	4
q_0	0.95	0.90	0.85	0.80

ρ	0.01	0.05	0.1	0.15
Ant size	0.5*Fleet size= 120	Fleet size=240	1.5*Fleet size=360	2*Fleet size=480

Table 5: Parameters levels of the lower-level ACO.

Parameter	Level 1	Level 2	Level 3	Level 4
α'	1	2	3	4
β'	1	2	3	4
q_0'	0.95	0.90	0.85	0.80
ρ'	0.01	0.05	0.1	0.15
Ant size	0.5*number of maintenance operations=30	number of maintenance operations=60	1.5*number of maintenance operations=90	2*number of maintenance operations=120

After selecting the most important parameters and their levels, now, it is the turn for selecting the best setting for these parameters. For this purpose, Taguchi method is adopted because it is one of the powerful tools for identifying the best setting of the parameters using orthogonal array technique and signal-to-noise (*S/N*) ratios (Dehnad, 1989). For the orthogonal array technique, it is an economic design technique that is used to minimize the number of experiments cases. Regarding the *S/N* ratio, it can be considered as a performance indicator that reflects the quality of each experiment case. Using Taguchi method to handle five parameters with four levels, as in the upper and lower-level ACO, leads naturally to select L_{16}' to be our orthogonal array in the Taguchi experiment design. Since we have two levels of ACO, the Taguchi experiment design was conducted twice using Minitab software. The average *S/N* ratio of the parameters at each level, while handling the upper and lower-level ACO, are illustrated in Fig. 8 and Fig. 9, respectively. Since the objective function in our study is to minimize the cost, the parameter level should be selected based on the smaller is better, meaning that a level with a small mean *S/N* is superior to the one with higher mean. Applying this rule in Fig. 8 and Fig. 9 results in selecting the best levels of the parameters as shown in Table 6. Since parameters Q and Q' are not among the most important parameters that influence the ACO performance, we assume that these parameters take the value of 0.01.

Insert Fig. 8 appropriate here

Insert Fig. 9 appropriate here

Table 6: Best parameter setting for the upper-level ACO and lower-level ACO.

Upper -level ACO		Lower-level ACO	
Parameter	**Best level**	**Parameter**	**Best level**
α	1	α'	1
β	2	β'	1
q_0	0.95	q_0'	0.95
ρ	0.10	ρ'	0.15
Ant size	240	Ant size	60

8.4 Solution of leader-follower Stackelberg game model

In this section, the bi-level nested ACO algorithm is implemented in order to obtain near optimal solutions for the joint SOAMRPFD and MSP, which is formulated as a leader-follower Stackelberg game. It is noteworthy that our algorithm is implemented based on the best parameter setting determined in the previous section.

Fig. 10 shows the tradeoffs between the upper-level ACO for the expected propagated delay cost of the airline and the lower-level ACO for the labor cost of the maintenance company. It is clear from Fig. 10 that the upper-level ACO reaches the convergence point and gives its best result after 350 iterations. On the other hand, after 450 iterations, the lower-level ACO reaches the convergence point and its best result is achieved. Since these results (at the convergence points) constitute the situation in which both companies are unwilling to change their decisions and objective functions, the Stackelberg equilibrium is achieved, with values 1,635.23 for the airline and 23,192.53 for the maintenance company.

After presenting the results of our proposed bi-level nested ACO algorithm, it is clear that it can handle a case study, which its upper-level optimization model consists of 240 flight legs and 4 maintenance stations, and its lower-level optimization model consists of nearly 60 maintenance operations for aircraft and 360 potential team sizes. As mentioned earlier, it was reported by Wang et al. (2016) that the direct methods such as satisfactory solution methods proposed by Muñoz & Abdelaziz (2012), could not solve their case study, which consists of 10 power transformer base modules, 32 suppliers, 6 manufacturers, 4 assembly plants, and 3 distribution centers. From the above observation, it is obvious that our proposed algorithm can efficiently handle a case study, which its size is much larger compared to the case study by Wang et al. (2016) that could not be handled by satisfactory solution methods. Therefore, our proposed algorithm is more efficient to handle large-scale problems compared to satisfactory solution methods.

Insert Fig. 10 appropriate here

8.5 Performance of the Stackelberg game model and the existing method

The performance of the developed Stackelberg game decision model (LFS) has been given in the previous section. Presenting the LFS performance is not adequate in demonstrating its advantage over the existing methods in the literature. Accordingly, the computational experiments are extended with the aim of comparing the LFS performance with the non-joint optimization method (NJOP), which is one of the traditional methods that is commonly used in the literature. The NJOP appeared in the study by Liang et al. (2015) that solved OAMRPFD independently from MSP, and also noticed in the study by Yan et al. (2004) that handled MSP in isolation from OAMRPFD.

In particular, the NJOP method treats each problem of SOAMRPFD and MSP as individual optimization problems and returns separate solutions for each. It first optimizes SOAMRPFD solely on the minimal cost, and then optimizes MSP to minimal cost as well. The results of these computational experiments, while handling SOAMRPFD and MS separately, are presented in Fig. 11 and Fig.12, respectively.

In terms of the expected propagated delay cost of the airline, Fig. 11 shows that the LFS model outperforms the NJOP model by 15.72% (1,635.23 vs. 1,940.20). Similarly, Fig. 12 shows further outperformance of the LFS model over the NJOP model by 18.56% (23,192.53 vs. 28,480.78), while handling the labor cost of the maintenance company. The rationale behind this outperformance lies in the fact that the LFS model

optimizes SOAMRPFD in conjunction with MSP, which means that the results of one part are sent back to the other part. This results in giving the two parties a chance to continue adjusting their decisions in order to improve the results until reaching equilibrium. In contrast, the NJOP model optimizes both problems separately, which means there is no feedback movement, leading to a loss of opportunity in adjusting the decisions and improving the obtained results.

So, it is clear cut from this section that the proposed LFS model significantly improves the results obtained by the airline and maintenance company. This echoes the importance of the coordinated configuration of SOAMRPFD and MSP being implemented in reality.

Insert Fig. 11 appropriate here

Insert Fig.12 appropriate here

8.6 Performance of the bi-level nested ACO algorithm

In the previous sub-sections, we present the performance of the developed Stackelberg game decision model using the bi-level nested ACO algorithm. In fact, solving a single test instance, as in our case study, is not enough to discuss the efficiency of the proposed algorithm, however, with this test instance, we are able to compare the performance of the developed Stackelberg game model with the non-joint optimization method. Therefore, to assess the applicability and scalability of the proposed algorithm, the computational experiments are further extended using test instances with different sizes. These experiments include using another three SOAMRPFD test instances, and the same MSP test instance presented in our case study. It should be noted here that the additional SOAMRPFD test instances are provided by the same airline that provides data for our case study. Detailed information about the test instances are shown in Table 7. For all test instances, it was assumed by the airline that T_{max} is 40 hours, MAT is 8 hours, TRT is 45 minutes, and C_{pD}follows the same information presented in Table 3.

Table 7: Characteristics of the test cases

Test case	Airline (SOAMRPFD)					Maintenance company (MSP)
	I	KT	C_{max}	A	MT	
Case 1	160	11	7	5	4	The information is same as Table 3
Case 2 (Our case study)	240	30	10	8	4	
Case 3	296	35	10	26	4	
Case 4	400	42	10	28	4	

Table 8 summarizes the results obtained from the bi-level nested ACO algorithm, as shown in the first four columns of the Table. The $Z_{SOAMRPFD}$ and Z_{MSP} columns represent the Stackelberg equilibrium for airline and maintenance companies, respectively. The No. of runs column shows the number of runs the algorithm takes to get the Stackelberg equilibrium, whereas the $CPU(min)$ column records the computational time of the algorithm as obtained from the MATLAB's internal calculation function. To assess the performance of the proposed algorithm, we calculate the approximated lower bound for SOMARPFD and MSP using the algorithms explained in section 7.4. These bounds and their computational times are reported in columns 5

to 8 of Table 8. After calculating the lower bounds, we calculate the gap ($\%GAP$) between the solution obtained from the bi-level nested ACO algorithm and the lower bound, to be the criterion for evaluating the algorithm performance. These gaps are shown in the last two columns of Table 8

Table 8: Performance characteristics of Bi-level nested ACO algorithm

Test case	Bi-level nested ACO algorithm				Approximated lower bound				$\%GAP$=(Z-*LB*) *100/Z	
	$Z_{SOAMRPFD}$	Z_{MSP}	No. of runs	$CPU(min)$	SOAMRPFD		MSP		SOAMRPFD	MSP
					$LB_{routing}$	$CPU(min)$	$LB_{staffing}$	$CPU(min)$		
Case 1	0	6800.54	1	20.45	0	600.15	6800.54	0.66	0	0
Case 2	1635.40	23192.53	1	66.67	1615.36	1158.25	23098.87	0.82	1.23	0.40
Case 3	825.85	25250.69	1	85.45	819.52	1947.72	25119.53	1.29	0.77	0.52
Case 4	1125.42	33450.23	1	115.85	1112.87	2689.24	33192.48	1.42	1.12	0.77

A close look at the results in Table 8, we can see that the bi-level nested ACO algorithm produces high quality solutions in a reasonable computational time. Regarding the solution quality, the high quality is apparent in both of SOAMRPFD and MSP. Starting with SOAMRPFD, it is noted that $Z_{SOAMRPFD}$ reaches $LB_{routing}$, as in the first case. As the number of flight legs and aircraft increase, $Z_{SOAMRPFD}$ deviates from $LB_{routing}$ with a $\%GAP$ of at most 1.23%, as in the second case. Moving to Z_{MSP}, it also reaches the $LB_{staffing}$, as in case 1, but when the number of maintenance operations increases, Z_{MSP} deviates slightly from $LB_{staffing}$ by at most 0.77%, as in the last case. With respect to the computational time, the bi-level nested ACO algorithm shows a reasonable performance. This is clearly shown in case 4 since the proposed algorithm produces the solution in about 2 hours, while the approximated lower bound needs up to 2 days. The reason behind this long computational time, is due to the large number of feasible routes that could be generated while computing the approximated lower bound of SOAMRPFD. Finally, we can see that getting the approximated lower bound is computationally expensive, as we need to generate all feasible routes, but, using the bi-level nested ACO algorithm can solve this issue with a small deviation from the approximated lower bounds. Therefore, it is clear from this section that the proposed bi-level nested ACO algorithm can efficiently handle cases with different sizes, as it produces profitable solutions in a reasonable computational time.

9. Conclusions and future directions

In this research work, we propose a joint optimization model for coordinated configuration of SOAMRPFD and MSP utilizing the Stackelberg game. In this game, SOAMRPFD, which is handled by the airline, plays the leader's role for minimization the propagated delay cost. On the other hand, MSP, which is handled by the maintenance company, performs the role of the follower that reacts rationally to the leader's decision with respect to the departure time of the airline's aircraft from the maintenance company. In order to achieve the Stackelberg equilibrium, a nested ACO algorithm is applied to provide a solution to the joint optimization model.

The results from a case study of a Middle Eastern major airline and a maintenance company validate the superiority of the proposed model. The results demonstrate significant savings in the costs of both companies compared to the results obtained from the traditional non-joint optimization method. Furthermore, they indicate that the proposed model has great potential for implementation in actual practice.

This work presents a formulation for a unique problem in the literature, whereas the case study verifies the effectiveness of the proposed work. However, there are some limitations that are suggested for future works.

The proposed MSP model assumes that the workforce capacity is deterministic. In this regard, relaxing this assumption is suggested for future research. Further, the proposed SOAMRPFD has a scope that is limited to four days. It would be beneficial to extend the scope to weekly, in which the number of flights and aircraft increase significantly.

Acknowledgments

The work presented in this paper was supported by grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. PolyU 15201414). In addition, this work was received a support from different grants, including; The Natural Science Foundation of China (Grant No. 71471158); The Research Committee of Hong Kong Polytechnic University (Project Numbers G-YBFD; G-UA4F; G-YBN1); and The Hong Kong Polytechnic University under student account code RTYN.

References

Aust, G., & Buscher, U. (2012). Vertical cooperative advertising and pricing decisions in a manufacturer–retailer supply chain: A game-theoretic approach. *European Journal of Operational Research, 223*(2), 473-482. doi:http://dx.doi.org/10.1016/j.ejor.2012.06.042

Balseiro, S. R., Loiseau, I., & Ramonet, J. (2011). An Ant Colony algorithm hybridized with insertion heuristics for the Time Dependent Vehicle Routing Problem with Time Windows. *Computers & Operations Research, 38*(6), 954-966. doi:http://dx.doi.org/10.1016/j.cor.2010.10.011

Bard, J. F., & Falk, J. E. (1982). An explicit solution to the multi-level programming problem. *Computers & Operations Research, 9*(1), 77-100. doi:http://dx.doi.org/10.1016/0305-0548(82)90007-7

Başdere, M., & Bilge, Ü. (2014). Operational aircraft maintenance routing problem with remaining time consideration. *European Journal of Operational Research, 235*(1), 315-328. doi:http://dx.doi.org/10.1016/j.ejor.2013.10.066

Beaumont, N. (1997). Scheduling staff using mixed integer programming. *European Journal of Operational Research, 98*(3), 473-484. doi:http://dx.doi.org/10.1016/S0377-2217(97)00055-6

Beli, J., #x00EB, Cardoen, B., & Demeulemeester, E. (2012). Improving Workforce Scheduling of Aircraft Line Maintenance at Sabena Technics. *Interfaces, 42*(4), 352-364. doi:10.1287/inte.1110.0585

Beliën, J., Demeulemeester, E., De Bruecker, P., Van den Bergh, J., & Cardoen, B. (2013). Integrated staffing and scheduling for an aircraft line maintenance problem. *Computers & Operations Research, 40*(4), 1023-1033. doi:http://dx.doi.org/10.1016/j.cor.2012.11.011

Chen, Z.-L., Li, S., & Tirupati, D. (2002). A scenario-based stochastic programming approach for technology and capacity planning. *Computers & Operations Research, 29*(7), 781-806. doi:http://doi.org/10.1016/S0305-0548(00)00076-9

Clarke, L., Johnson, E., Nemhauser, G., & Zhu, Z. (1997). The aircraft rotation problem. *Annals of Operations Research, 69*(0), 33-46. doi:10.1023/A:1018945415148

Dehnad, K. (1989). Quality control, robust design, and the Taguchi method: Wadsworth & Brooks/Cole Advanced Books & Software.

Deng, G. F., & Lin, W. T. (2011). Ant colony optimization-based algorithm for airline crew scheduling problem. *Expert Systems with Applications, 38*(5), 5787-5793. doi:http://dx.doi.org/10.1016/j.eswa.2010.10.053

Dietz , D. C., & Rosenshine , M. (1997). Optimal specialization of a maintenance workforce. *IIE Transactions, 29*(5), 423-433. doi:10.1023/a:1018560321220

Dorigo, M. & Stützle, T. (2004). Ant Colony Optimization: MIT Press, Cambridge, MA.

Dunbar, M., Froyland, G. & Wu, C.-L. (2012). Robust Airline Schedule Planning: Minimizing Propagated Delay in an Integrated Routing and Crewing Framework. Transportation Science, 46(2), 204-216.

Dunbar, M., Froyland, G., & Wu, C.-L. (2014). An integrated scenario-based approach for robust aircraft routing, crew pairing and re-timing. *Computers & Operations Research, 45*(0), 68-86. doi:http://dx.doi.org/10.1016/j.cor.2013.12.003

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

Eltoukhy, A.E.E., Chan, F.T.S., 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.E., Chan, F.T.S., Chung, S.H., Niu, B. & Wang, X.P. (2017b). 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.

Esmaeili, M., Aryanezhad, M.-B., & Zeephongsekul, P. (2009). A game theory approach in seller–buyer supply chain. *European Journal of Operational Research, 195*(2), 442-448. doi:http://dx.doi.org/10.1016/j.ejor.2008.02.026

Feo, T.A. & Bard, J.F. (1989). Flight Scheduling and Maintenance Base Planning. Management Science, 35(12), 1415-1432.

Gopalan, R., & Talluri, K. T. (1998). The Aircraft Maintenance Routing Problem. *Operations Research, 46*(2), 260-271. doi:10.1287/opre.46.2.260

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. doi:10.1287/trsc.1120.0433

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

Lan, S., Clarke, J.-P., & Barnhart, C. (2006). Planning for Robust Airline Operations: Optimizing Aircraft Routings and Flight Departure Times to Minimize Passenger Disruptions. *Transportation Science, 40*(1), 15-28. doi:10.1287/trsc.1050.0134

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. doi:doi:10.1287/trsc.1100.0338

Liang, Z., Feng, Y., Zhang, X., Wu, T., & Chaovalitwongse, W. A. (2015). Robust weekly aircraft maintenance routing problem and the extension to the tail assignment problem. *Transportation Research Part B: Methodological, 78*, 238-259. doi:http://dx.doi.org/10.1016/j.trb.2015.03.013

Mohammadi, S., Soleymani, S., & Mozafari, B. (2014). Scenario-based stochastic operation management of MicroGrid including Wind, Photovoltaic, Micro-Turbine, Fuel Cell and Energy Storage Devices. *International Journal of Electrical Power & Energy Systems, 54*, 525-535. doi:http://doi.org/10.1016/j.ijepes.2013.08.004

Muñoz, M. M., & Abdelaziz, F. B. (2012). Satisfactory solution concepts and their relations for Stochastic Multiobjective Programming problems. *European Journal of Operational Research, 220*(2), 430-442. doi:https://doi.org/10.1016/j.ejor.2012.01.052

Qin, Y. (2012). A stackelberg-game model in a two-stage supply chain. *Systems Engineering Procedia, 3*, 268-274. doi:http://dx.doi.org/10.1016/j.sepro.2011.11.029

Ren, A., & Wang, Y. (2016). A novel penalty function method for semivectorial bilevel programming problem. *Applied Mathematical Modelling, 40*(1), 135-149. doi:https://doi.org/10.1016/j.apm.2015.04.041

Samimi, A., Nikzad, M., & Siano, P. (2017). Scenario-based stochastic framework for coupled active and reactive power market in smart distribution systems with demand response programs. *Renewable Energy, 109*, 22-40. doi:http://doi.org/10.1016/j.renene.2017.03.010

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. doi:http://dx.doi.org/10.1016/j.ejor.2004.10.033

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. doi:http://dx.doi.org/10.1016/S0965-8564(02)00004-6

Stackelberg, H. V. (1952). *The Theory of the Market Economy*. Oxford,U.K.: Oxford University Press.

Talluri, K. T. (1998). The Four-Day Aircraft Maintenance Routing Problem. *Transportation Science, 32*(1), 43-53. doi:10.1287/trsc.32.1.43

Tsao, Y.-C., Lu, J.-C., An, N., Al-Khayyal, F., Lu, R. W., & Han, G. (2014). Retailer shelf-space management with trade allowance: A Stackelberg game between retailer and manufacturers. *International Journal of Production Economics, 148*, 133-144. doi:http://dx.doi.org/10.1016/j.ijpe.2013.09.018

van Hoesel, S. (2008). An overview of Stackelberg pricing in networks. *European Journal of Operational Research, 189*(3), 1393-1402. doi:http://dx.doi.org/10.1016/j.ejor.2006.08.064

Wang, D., Du, G., Jiao, R. J., Wu, R., Yu, J., & Yang, D. (2016). A Stackelberg game theoretic model for optimizing product family architecting with supply chain consideration. *International Journal of Production Economics, 172*, 1-18. doi:https://doi.org/10.1016/j.ijpe.2015.11.001

Wu, Z., Zhao, N., Ren, G., & Quan, T. (2009). Population declining ant colony optimization algorithm and its applications. *Expert Systems with Applications, 36*(3, Part 2), 6276-6281. doi:http://dx.doi.org/10.1016/j.eswa.2008.07.013

Xiao, T., Choi, T.-M., & Cheng, T. C. E. (2014). Product variety and channel structure strategy for a retailer-Stackelberg supply chain. *European Journal of Operational Research, 233*(1), 114-124. doi:http://dx.doi.org/10.1016/j.ejor.2013.08.038

Yan, S., Yang, T.-H., & Chen, H.-H. (2004). Airline short-term maintenance manpower supply planning. *Transportation Research Part A: Policy and Practice, 38*(9–10), 615-642. doi:http://dx.doi.org/10.1016/j.tra.2004.03.005

Yang, T.-H., Yan, S., & Chen, H.-H. (2003). An airline maintenance manpower planning model with flexible strategies. *Journal of Air Transport Management, 9*(4), 233-239. doi:http://dx.doi.org/10.1016/S0969-6997(03)00013-9

Yen, J. W., & Birge, J. R. (2006). A Stochastic Programming Approach to the Airline Crew Scheduling Problem. *Transportation Science, 40*(1), 3-14. doi:10.1287/trsc.1050.0138

Yin, P.-Y., & Wang, J.-Y. (2006). Ant colony optimization for the nonlinear resource allocation problem. *Applied Mathematics and Computation, 174*(2), 1438-1453. doi:http://doi.org/10.1016/j.amc.2005.05.042

Yu, B., & Yang, Z. Z. (2011). An ant colony optimization model: The period vehicle routing problem with time windows. *Transportation Research Part E: Logistics and Transportation Review, 47*(2), 166-181. doi:http://dx.doi.org/10.1016/j.tre.2010.09.010

Yu, Y. & Huang, G.Q. (2010). Nash game model for optimizing market strategies, configuration of platform products in a Vendor Managed Inventory (VMI) supply chain for a product family. European Journal of Operational Research, 206(2), 361-373.

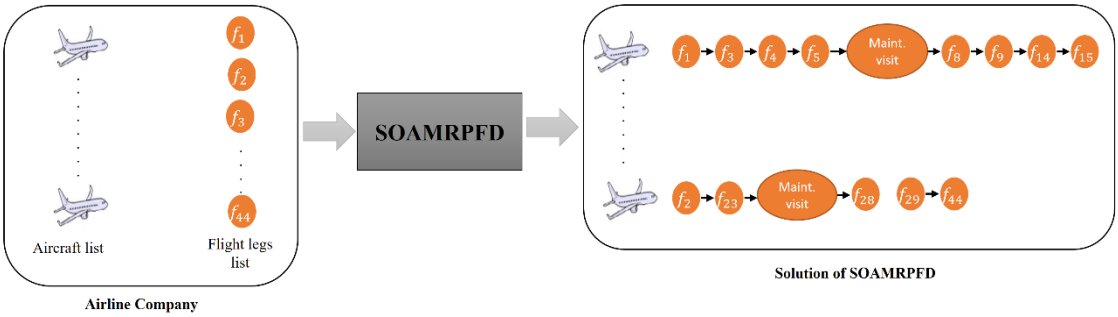

Fig. 1: Typical representation of the routing problem and its solution

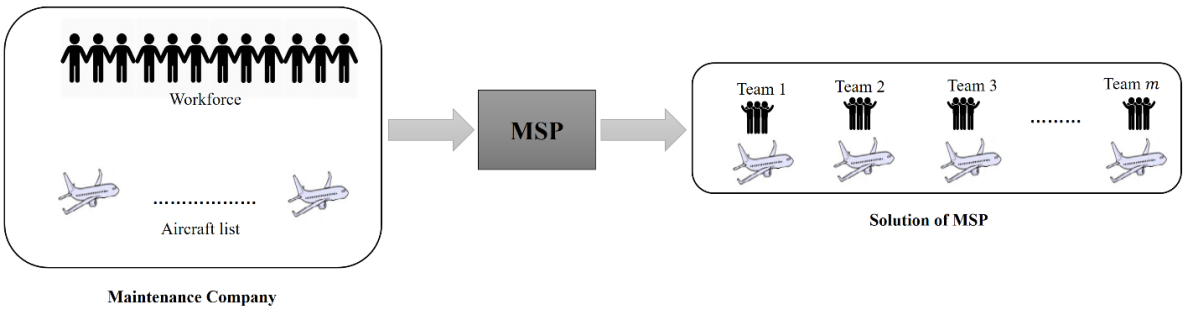

Fig. 2: Typical representation of the staffing problem and its solution

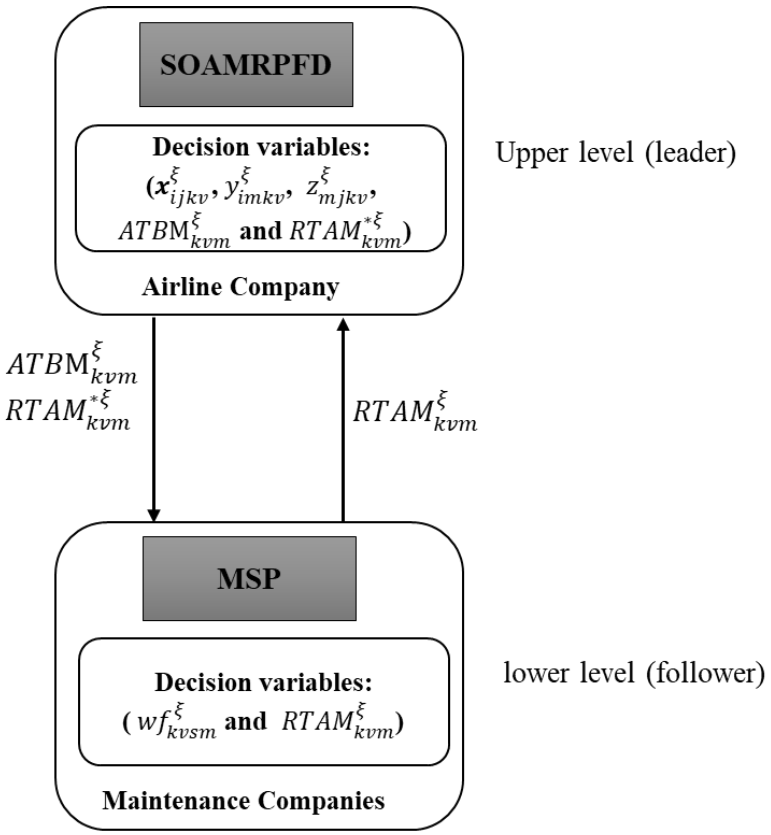

Fig. 3: Joint configuration for SOAMRPFD and MSP

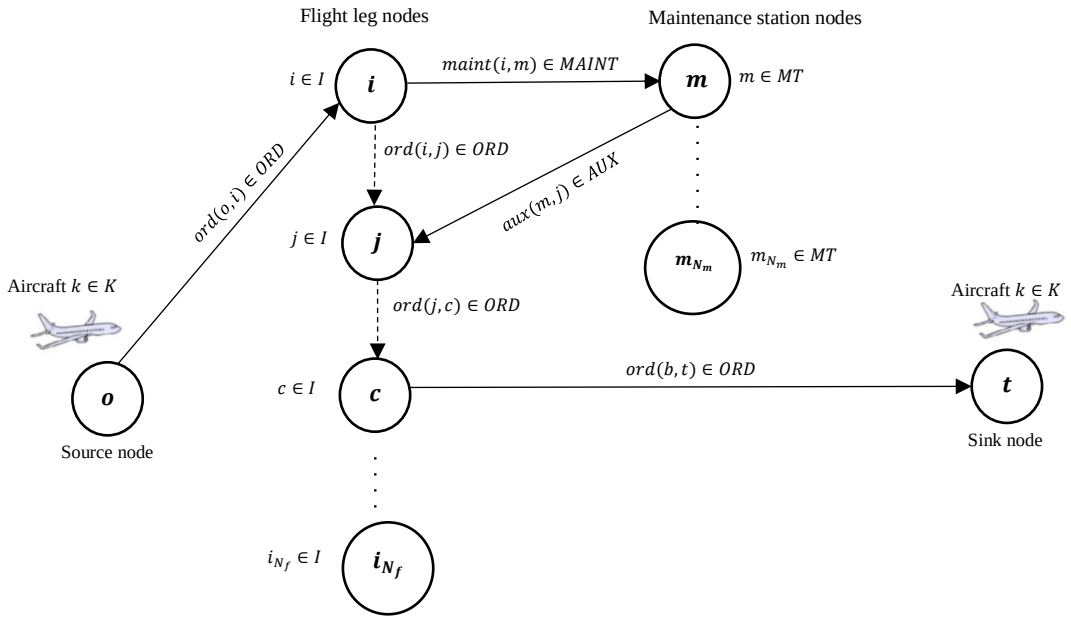

Fig. 4: Modified connection network representation.

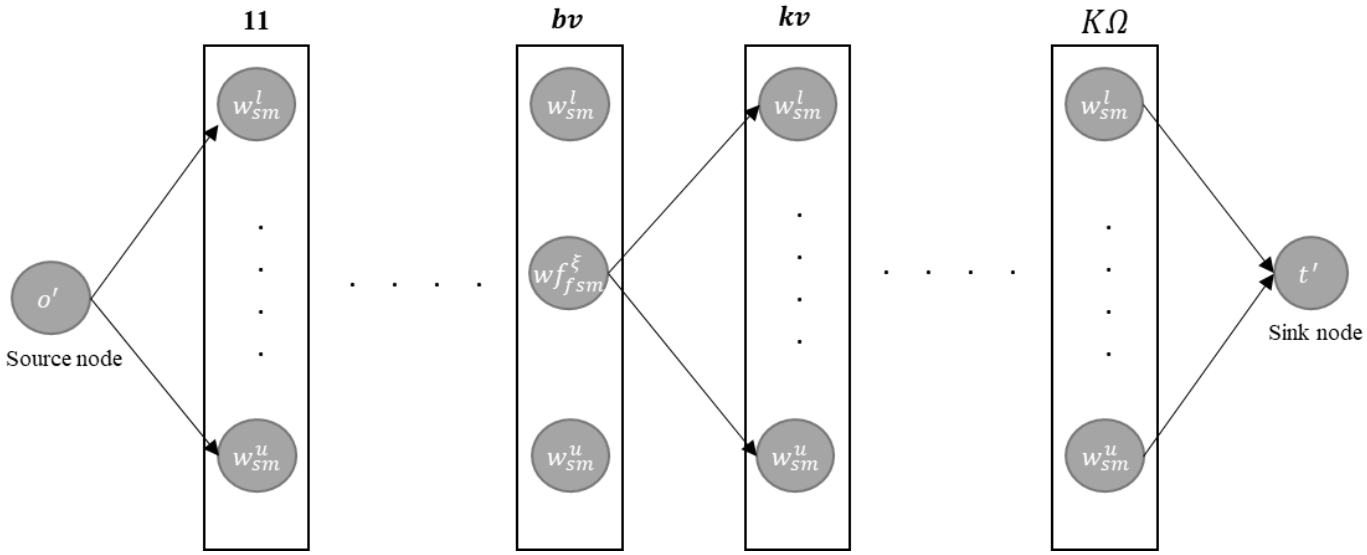

Fig. 5: MSP layered graph representation.

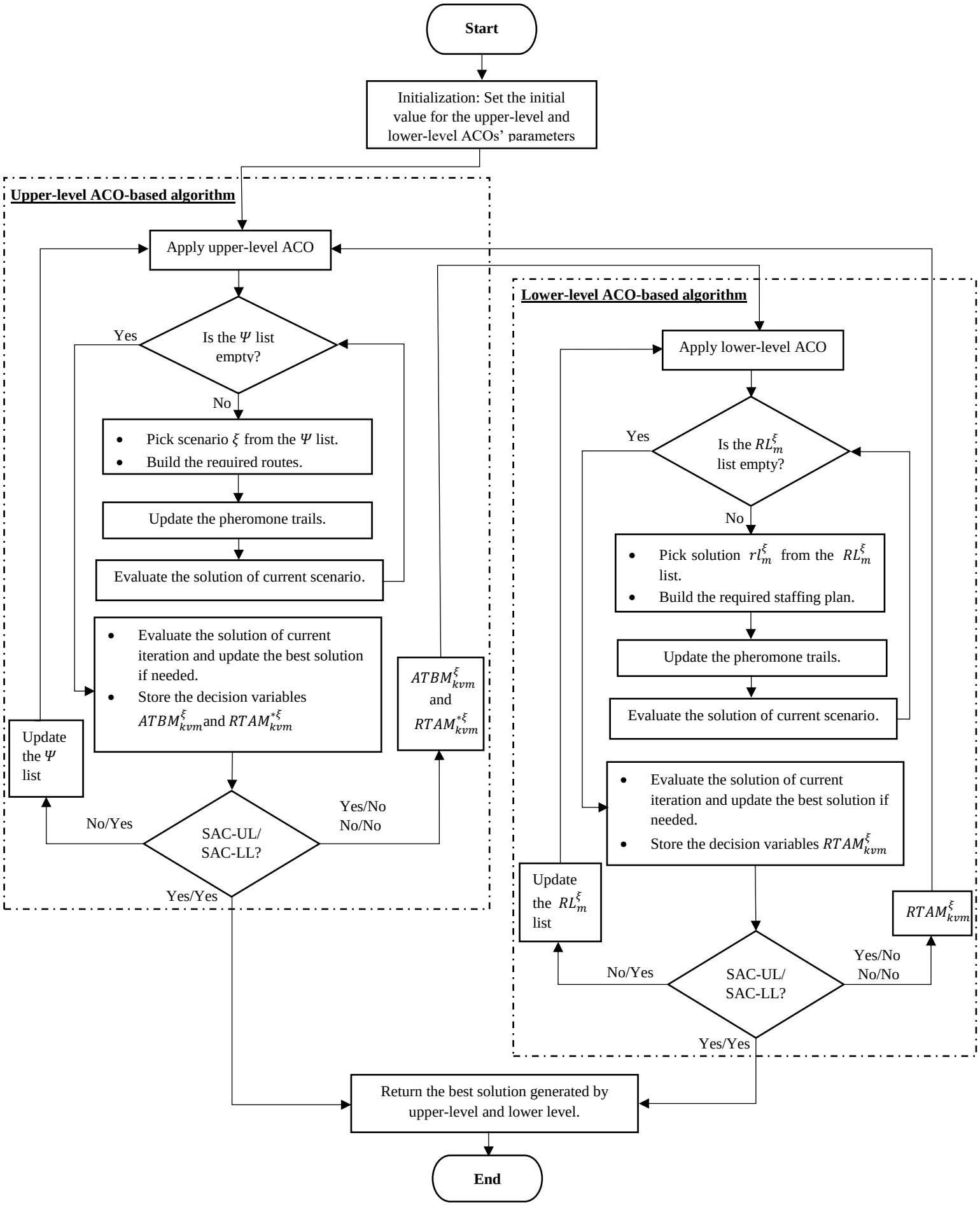

Fig. 6: Flow chart of the bi-level nested ACO algorithm.

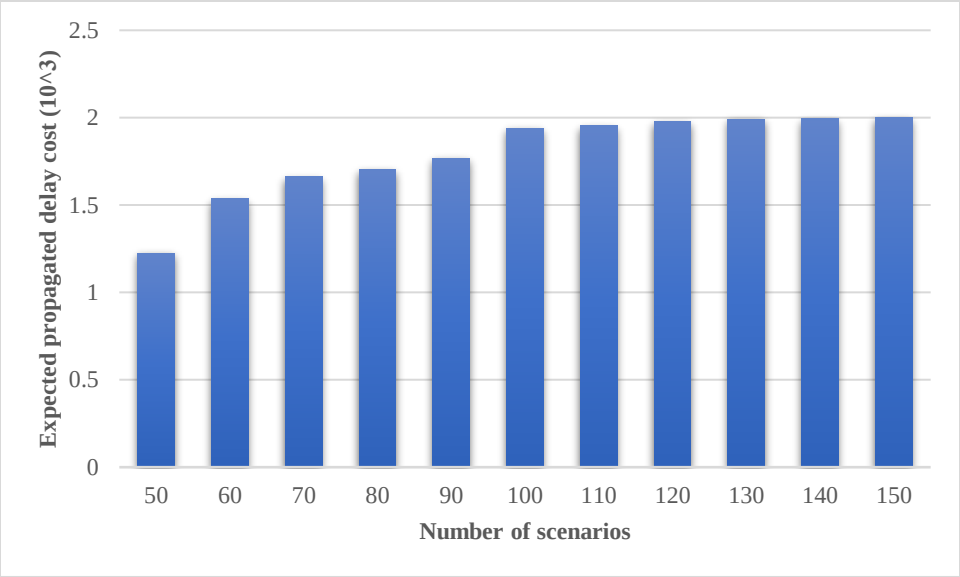

Fig. 7: Results of SOAMRPFD under different scenario numbers.

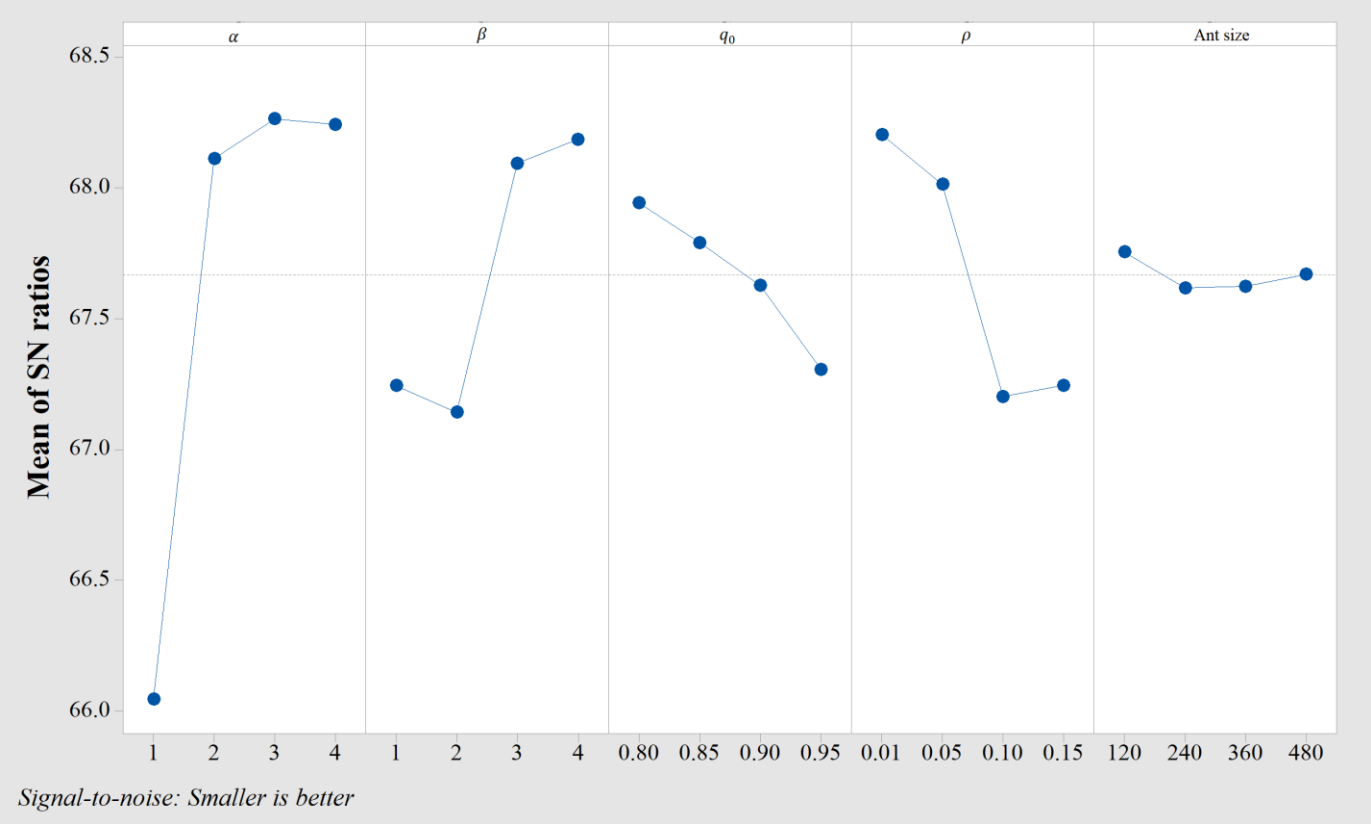

Fig. 8: Main effect plot for *S/N* ratio while using upper-level ACO algorithm

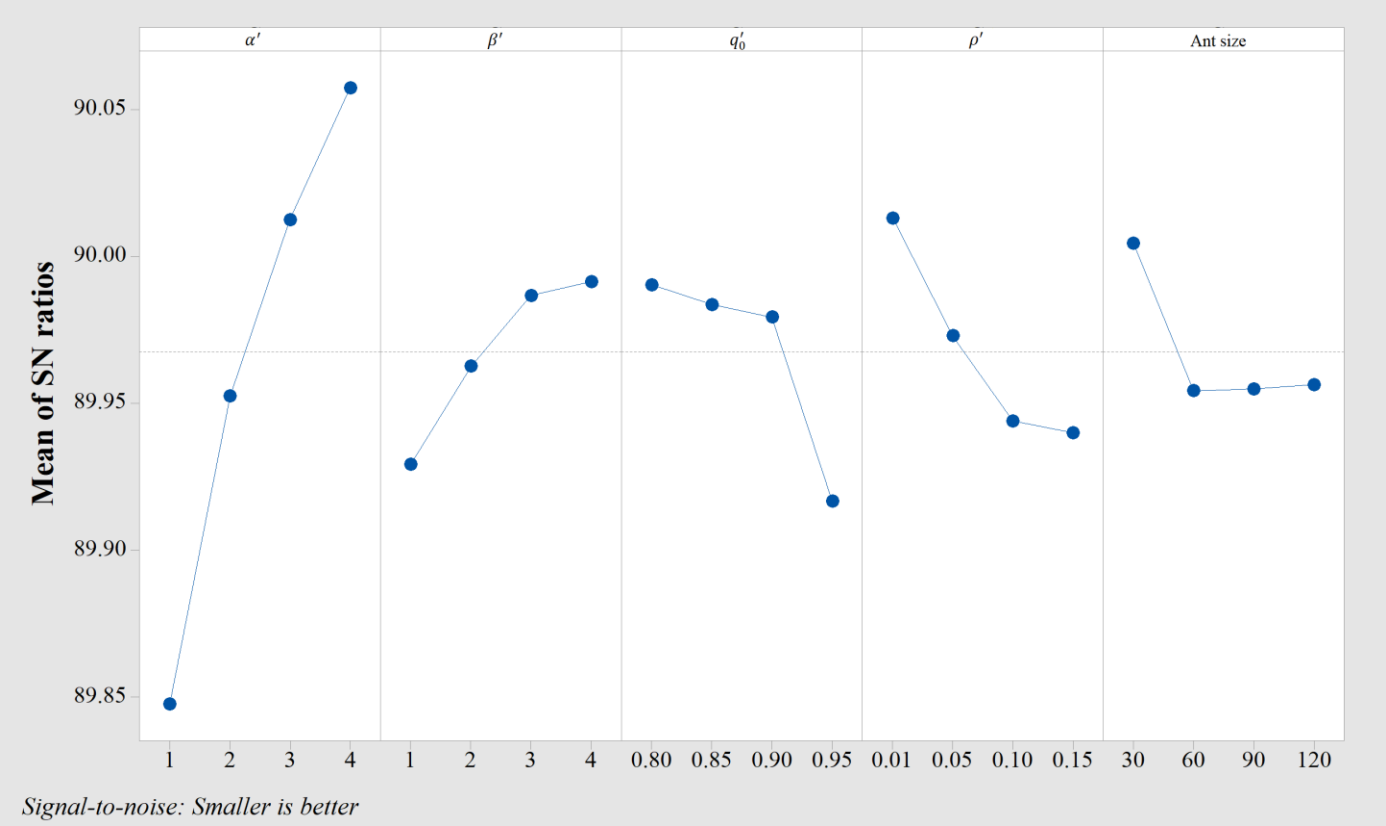

Fig. 9: Main effect plot for *S/N* ratio while using the lower-level ACO algorithm

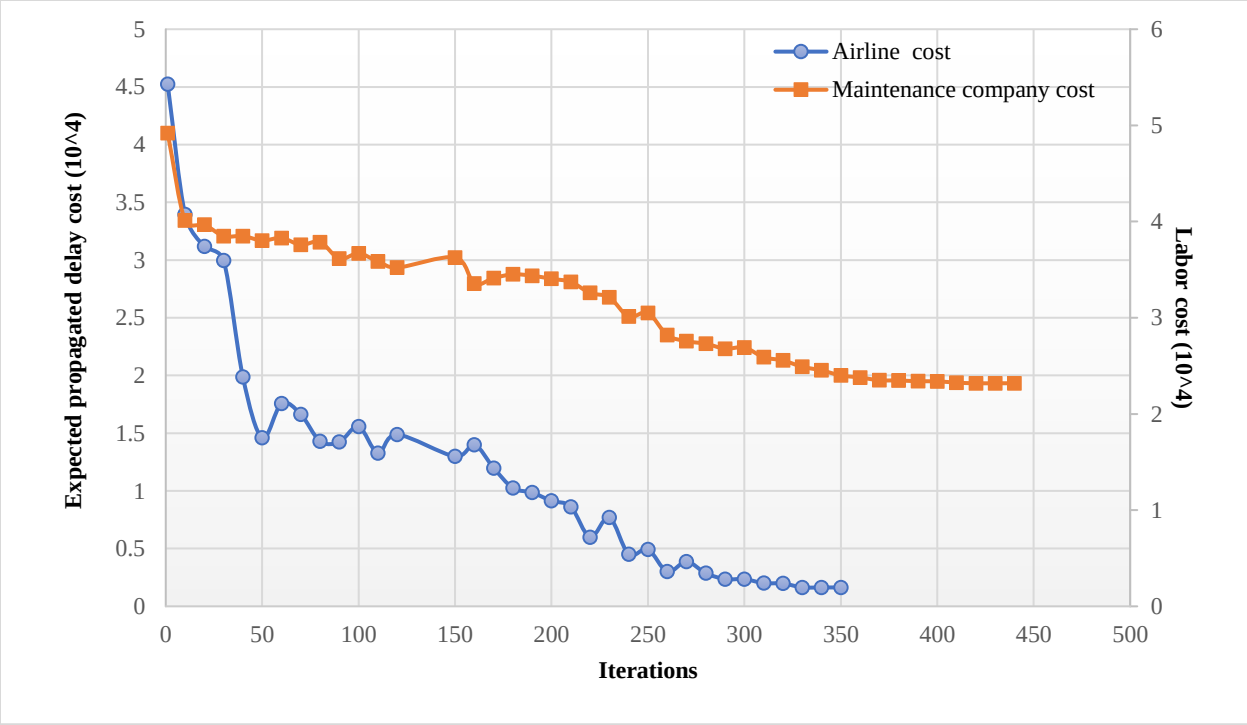

Fig. 10: Convergence of the nested ACO algorithm.

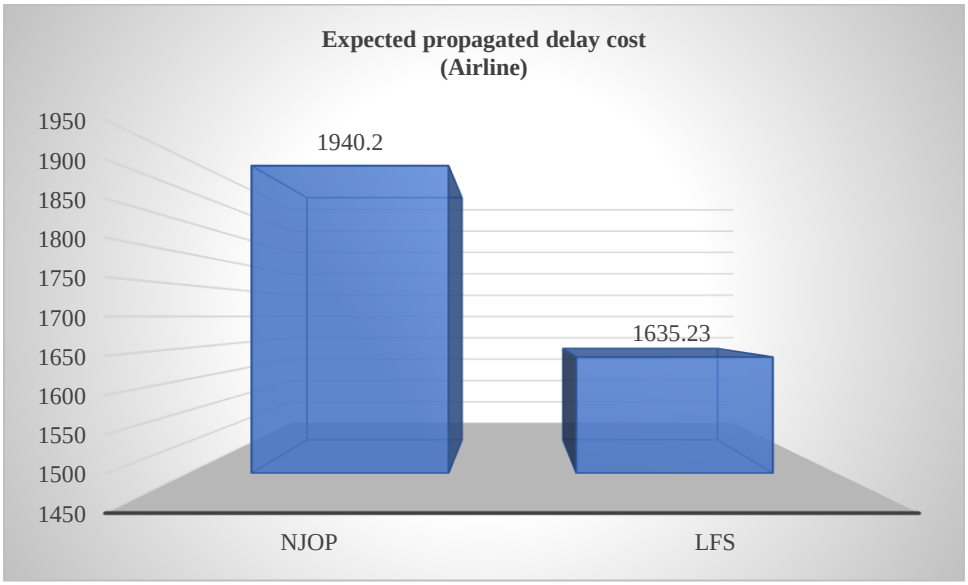

Fig. 11: Performance comparison for SOAMRPFD.

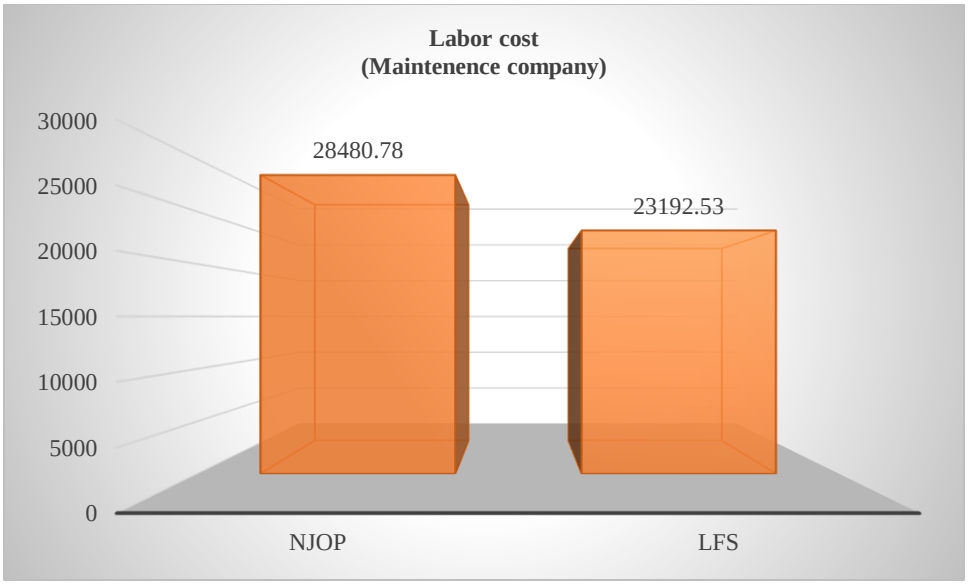

Fig. 12: Performance comparison for MSP.