MPRO 2012-2013

RAPPORT DE STAGE DE MASTER

Aircraft routing: complexity and algorithms

Axel Parmentier

MASTER PARISIEN DE RECHERCHE OPÉRATIONNELLE

Laboratoire d'accueil : CERMICS
Directeur de stage : Frédéric Meunier

Version revised
Champs sur Marne, November 20, 2013

Remerciements

Tout d'abord, je souhaite remercier l'équipe de Recherche Opérationnelle d'Air France, et en particulier Sylvain Le-Nestour, Cyrille Szymanski et Christophe Ressel pour le temps qu'ils m'ont consacré et pour la vue d'ensemble des applications et enjeux de la recherche opérationnelle appliquée au secteur du transport aérien qu'ils ont pû me donner.

Enfin et surtout, je souhaite remercier chaleureusement Frédéric Meunier pour sa disponibilité et pour la qualité de son encadrement, de son suivi et de ses conseils tout au long de ce stage, qui annoncent une collaboration passionnante en thèse ces trois prochaines années.

Contexte du stage

Je commence en septembre 2013 une thèse sur l'optimisation simultanée des rotations avions et des rotations équipages, dont le but sera de réunir en un seul problème d'optimisation *l'aircraft routing problem* et le *crew pairing problem* étudiés dans ce rapport. Cette thèse sera réalisée au CERMICS sous la direction de Frédéric Meunier, et en partenariat avec Air France. Le sujet de ce stage est donc un préliminaire théorique à ma thèse. En conséquence, la première partie de ce stage a consisté en une revue de littérature approfondie des problèmes de planning et de tournées avions et équipages auxquels les compagnies aériennes sont confrontées. Au bout d'un mois et demi, le sujet de la complexité de l'*aircraft routing problem* a été identifié comme pertinent pour être traité pendant la durée restante du stage. En conséquence, la structure de ce rapport correspond aux deux périodes de ce stage.

Enfin, la communauté scientifique de la recherche opérationnelle appliquée au secteur aérien travaillant en langue anglaise, nous avons choisi de rédiger ce rapport en anglais. La suite de ce rapport est donc rédigée sous la forme d'un article scientifique en Recherche Opérationnelle.

Abstract

The contribution of this paper is two-folds: first, an extensive literature review of the applications of Operations Research to air transportation schedule problems is proposed. Then, the complexity of *Aircraft Routing Problem* is studied. Aircraft Routing Problem is proved to be *NP-complete* in the general case. A *polynomial algorithm* is given to solve the Aircraft Routing Problem when fleet size is fixed. Finally, a compact linear program to solve aircraft routing is introduced. The special purpose notions of *equigraph* and *network-state graph* are introduced to give a simple proof of these two results and to justify the validity of the linear integer program.

Keywords Aircraft Routing Problem – NP-completeness – Pebbling game.

Contents

Introduction

Operations research has been for sixty years a source of improvements in *Air Transport Industry*. Three main areas can be identified: the various stages of aircraft and crew schedule planning, revenue management, including over-booking and leg-based and network-based inventory management, and the planning and operations of aviation infrastructure (airports and air traffic management). For the first area, this paper provides a literature review and a brief summary of the state of the art in Section 1. One of the main stages of aircraft and crew schedule planning is known as the *aircraft routing problem*, which consists in assigning routes to airplanes to operate the flight schedule that respects maintenance conditions. The *NP-completeness* of aircraft routing feasibility problem and its polynomiality when the number of airplanes is fixed is proved in Sections 3.2 and 4. Finally a compact linear integer program to solve the aircraft routing problem is given in section 3.3.

1 Literature revue on applications of Operations Research to Air Transportation

An extensive review of applications of Operations Research to Air Transport Industry has been written by Cynthia Barnhart in 2003 [4]. In this section, we focus on aircraft and crew schedule planning. Schedule planning includes the design of the network of flights (which Origin-Destinations pairs flights will be covered and when), differing aircraft types, gate, airport slot and air traffic control restrictions, design of airplane routes and crew schedules etc. Several rules must be taken into account such as air traffic control restrictions, maintenance check for airplanes, work rules, use of gates and noise curfews etc. Due to this huge complexity, no *single* optimization model has been solved or even formulated for the whole problem. This resulted in the decomposition of the whole process in four steps operated by different divisions in companies. A set of corresponding subproblems often defined as follows have been identified in Operations Research.

1. *Schedule design:* Define which markets to serve and with what frequency, and how to schedule flights to meet these frequencies.
2. *Fleet assignment:* Specify what type of airplane to assign each flight leg.
3. *Aircraft routing:* Determining how to route airplanes to cover each flight leg with one and only airplane and to ensure maintenance requirements for airplanes.
4. *Crew scheduling:* Select which crews will cover each flight leg in order to minimize global crew costs.

The approach commonly used today is a sequential resolution of the subproblems, which gives a suboptimal but yet feasible solution. Each subproblem taken separately is already very rich, and a large literature has been developed in the last sixty years to solve these problems. New approaches combining two or three of the subproblems have raised recently, but the algorithms are still not mature enough to be used operationally. The section is organized as follows:

a brief description of solution methods used for separate subproblems is given, followed by a study of the new combined models. Finally, an overview of actual challenges is given.

1.1 Schedule design

Specifying the flight legs to be flown and the departure to be flown, the flight schedule largely defines the competitive position of a company. Designing a profit maximizing schedule is however extremely complex. Indeed, it is affected by other companies offers and time dependent Origin Destination demand, which is extremely difficult to capture. Thus, the typical practice today is to build flight schedules manually, with limited optimization. Nonetheless, recent research have been made by designing a simplified design problem involving only *incremental* changes to *existing flight schedules* [8, 28, 29]. Model integrating fleet assignment and incremental schedule optimization have recently been proposed [28].

1.2 Fleet assignment

Once the flight schedule is determined, the *fleet assignment problem* consists in finding the cost minimizing assignment of airplane types to flight legs. Fleeting costs are *operating costs*, specified for each (flight leg, airplane type) couple and representing the cost of operating one leg with one airplane, and *spill costs*, which measure the revenue lost when passenger demand for a flight leg exceeds the assigned airplane's seating capacity. Fleeting problems truly faced by airlines can today be solved efficiently thanks to a multicommodity flow formulation [1, 20]. Vertices represent time and location of flight leg departure and arrival. Arcs correspond to flights or ground arcs (representing airplanes on the ground between flights). There is one commodity for each airplane type. Fleet capacity and cover (each flight leg has to be covered exactly once in a feasible solution) are added. The next generation of models has introduced itinerary or Origin-Destinations based fleet assignment approaches, that capture the specific origin destination demand (for passengers with connections) [6, 22, 32].

1.3 Aircraft routing

Schedule design produces the flight network. Fleet assignment decisions decompose the network into subnetworks, each one associated with airplanes of a single type. The next step is *aircraft routing*. The goal is to determine *routing* or *rotations* for each airplane in a fleet. A routing is a sequence of flight covered by a single airplane. A rotation is a routing that starts and ends at the same location. Each airplane has to visit a maintenance station at regular intervals. Maintenance are performed in several airports called base at night. Each airplane has to spend a night in a maintenance base every D days. An *individual* airplane is assigned to each flight in a subnetwork. Traditional approach split the problem into two subproblems. First problem produces routing for one day, and second problem combines these one day origin destination routing solutions into a week long (or month long) routing which respects the maintenance requirement. A polynomial algorithm for the second subproblem exists for $D = 3$

[19], but for $D \geq 4$ problem is NP-complete [35]. Heuristics [16] and Lagrangian relaxation [10] have been proposed to solve the global problem.

As the cost of operating a flight with an airplane of a specific fleet is independent from the individual airplane, airplane routing has long been considered as a feasibility problem – and no optimization is today realized in most companies. Nonetheless, it has recently been linked to optimization for two different kinds of reasons. The first kind of reasons is its integration with the fleet assignment problem or the crew scheduling problem. In low frequency point to point networks, aircraft routing in often unfeasible given the fleet assignment solution. Researchers have therefore integrated fleet assignment and aircraft routing models [5, 14]. Integration of aircraft routing and crew scheduling is further discussed in Section 1.6. The second kind of reasons is delay. Indeed, delays on flights is a huge source of costs for airline companies. Statistical treatment shows that delay is more likely to arise on some specific flights. Thus aircraft routing can be optimized in order to minimize the expected value of total delay [25] – or any other probabilistic measure of delay.

Complexity of aircraft routing feasibility problem is further studied in Section 2.

1.4 Crew scheduling

Crew scheduling problems with numerous and complex work rules is a field where optimization is essential: the huge number of possible decisions make it extremely difficult to find feasible – let alone optimal – solutions manually. Moreover, crews represent airlines' second highest operating cost after fuel, thus even slight improvements in their utilization can translate into significant savings. Crew scheduling problem has thus been intensively studied ([18] for a state of the art). Even today, the crew scheduling problem is broken into two sequentially solved subproblems, the *crew pairing* problem and the *crew assignment problem*. The *crew pairing problem* generates minimum-cost, multiple-day work schedules called *pairings*. Regulatory agencies and collective bargaining agreements specify the many work rules that define how the flights legs can be combined to create feasible pairings. The cost structure of pairings address further complexity, and is typically represented as a nonlinear function of flying time, total elapsed work time, rest time, and total time away from the base. The *crew assignment problem* combines these pairings into equitable and efficient month-long crew schedules, called *bidlines* or *rosters* and assign them to individual crew members, taking into account their particular needs and requests. Later in this section, we focus on crew pairing.

Because of the complexity of the costs, obtaining an integer linear programming formulation of the problem is already difficult. Resource models give a flexible framework to take into account changing regulation [13]. Real life instances of the crew pairing problem have a huge size. Thus, heuristics have long been used to generate feasible solutions [2, 21]. To generate crew solutions with known optimal bound, researchers have turned to branch-and-price and column generation techniques [2, 3, 7, 9, 13, 21, 23, 24], [26, 27, 36]. *Branch-and-follows-on* branching strategies based on flight legs pairs, with one flight immediately following the other, are adopted. Even with this branching strategy, the number of branching decisions to prove optimality is typically excessive and branching heuristics are used.

1.5 Ongoing and future challenges

Notwithstanding the substantial progress made in solving aircraft and crew scheduling problems in recent decades, significant works remain. Beyond the progress to be made in the four problems taken separately, areas of research comprise: integrating the four problems, expand schedule planning models to include pricing and revenue management decisions, assessing the system-wide cost and service impact of paradigm shifts in airline scheduling, such as depeaking flight schedules by spreading out airplane arrival and departure banks, and develop stochastic tools to be able to take into account the robustness of operations with respect to delay. A developing and linked field of research is operations recovery: optimized solutions are rarely executed as planned, which tackles with unplanned disruptive events and attempts to minimize *realized* and non planned costs [10, 33, 34].

1.6 Integrating aircraft routing and crew scheduling

In this section, a brief overview of the growing literature on integrated aircraft routing and crew pairing problems is given. Researchers had the idea to reverse the order in which the last two problems are solved [24]. Indeed, as only feasibility is searched in aircraft routing problem whereas optimization is searched in crew pairing problem, solving crew pairing problem first enables to minimize the cost on a larger set of feasible pairings and thus achieve solutions of better quality. But this procedure often leads to infeasiblity in aircraft maintenance routing problem. Indeed, when the time between the arrival and the departure of a crew form an airport is too short, the two flights must be flown by the same airplane. Thus, if the crew pairing problem is solved first, all flights linked by a short connect have to be flown by the same airplane in the aircraft routing problem, which can lead to infeasibility. Thus came the idea of integrating crew pairing and aircraft routing problem. The integrated models takes the fleet specific schedule as input and finds the optimal solution on the set of feasible aircraft routing and crew paring couples.

Two lines of models have been developed for the integrated models. In the first line, aircraft routing solutions are generated first and included in an extended crew pairing problem [11]. Authors showed that only solutions including unique and maximal short connection sets needed to be generated to obtain optimal solutions. Matheuristics relying on branch and price are then used to solve the problem. The other line of models develops a three phase mathheuristics relying on Benders Decomposition [12]. The model is then generalized to integrate fleet assignment and restricted connect concept [30]. Finally, the efficiency of the algorithm is improved by a factor 10 by exchanging the master problem and the subproblem in the Benders decomposition [31]. Benders decomposition approach is more flexible, as extended crew pairing problem cannot integrate flight assignment and restricted connects. On medium scale instance, extended crew pairing method is faster than Benders decomposition method. Nonetheless, the time consuming phase of this algorithms is the generation of unique and maximal connection sets, and the time necessary to its resolutions grows fast. Thus, on large instances, Benders decomposition method gives better results [30].

The state of the art techniques still rely on matheuristics and do not solve

the integrated models to optimality. Moreover, algorithms are still not efficient enough to be used operationally. Nonetheless, they prove that significant savings can be made by integrating aircraft routing, crew pairing and fleet assignment models.

2 Paths partitions and aircraft routing problem

Aircraft routing problem is often referred to be NP-complete, but the mathematical formulation of the aircraft routing problem is not fixed, and we are not aware of a paper stating a precise definition of the problem completed by the proof of NP-completeness. In the remainder of this paper, we give a rigorous formulation of aircraft routing problem, we prove its NP-completeness in the general case and its polynomiality if fleet size k is fixed, and we give a integer linear program for aircraft routing.

2.1 Aircraft routing problem

The goal of the *aircraft routing problem* is to route airplanes while ensuring maintenance constraints. In the sequence of aircraft and crew schedule planning problems in air transport, aircraft routing comes after schedule design and fleet assignment. Thus, the set of flight legs operated by the company and the type of airplane which will cover each individual flight is already chosen. Thus, an aircraft routing problem is solved for each sub-fleet corresponding to the different types of airplane in the company. The aircraft routing problem is solved on a given horizon T, which is typically one week or one month long. Initial position of airplanes is given by the solution of the aircraft routing problem on the previous horizon. Each airplane's routing, i.e. the sequence of flight legs it covers during the horizon, must be determined. Regulation agencies impose regular maintenance stops for airplanes. If the long term maintenance checks (whose frequency is typically once a year) are not taken into account in the aircraft routing problem, the short term ones must be performed every 4 days, which is shorter than the aircraft routing horizon. These maintenance checks have thus to be taken into account in the routing problem. Their duration being quite long (typically 5 hours), these maintenance checks are performed at night. As the equipment and labor needed to perform these maintenance checks are expensive, only a few airports called *maintenance bases* are equipped. In the aircraft routing problem, each individual aircraft must stay one night in a maintenance base at least every D days (D is typically equal to 4).

Aircraft routing is solved on a period of time: *horizon* H is the number of days this period lasts. Time is discretized in τ steps per day. The network is composed of a set of *airports* $\alpha \in \mathcal{A}$, some of these being in the set of maintenance bases $\mathcal{B} \subseteq \mathcal{A}$. Each *flight* $f \in F \subseteq (\mathcal{A} \times [\tau] \times [H])^2$ is identified by two airport time day triples, one corresponding to its departure $(\alpha_f^{dep}, t_f^{dep}, d_f^{dep})$ and one corresponding to its arrival $(\alpha_f^{arr}, t_f^{arr}, d_f^{arr})$. The scheduled arrival time t_f^{arr} is equal to the scheduled landing time plus the minimum turn time between two flights – this way the airplane operating f can be chained to any flight leaving α_f^{arr} after t_f^{arr}. To each day $d \in [H-1]$ corresponds a night set $\Lambda_d \subseteq F \cup F^2$. A flight f belongs to night set Λ_d if $d_{f_1}^{dep} \leq d$ and $d_{f_2}^{arr} > d$. A couple of flights (f_1, f_2) belongs to night set Λ_d if the departure airport of

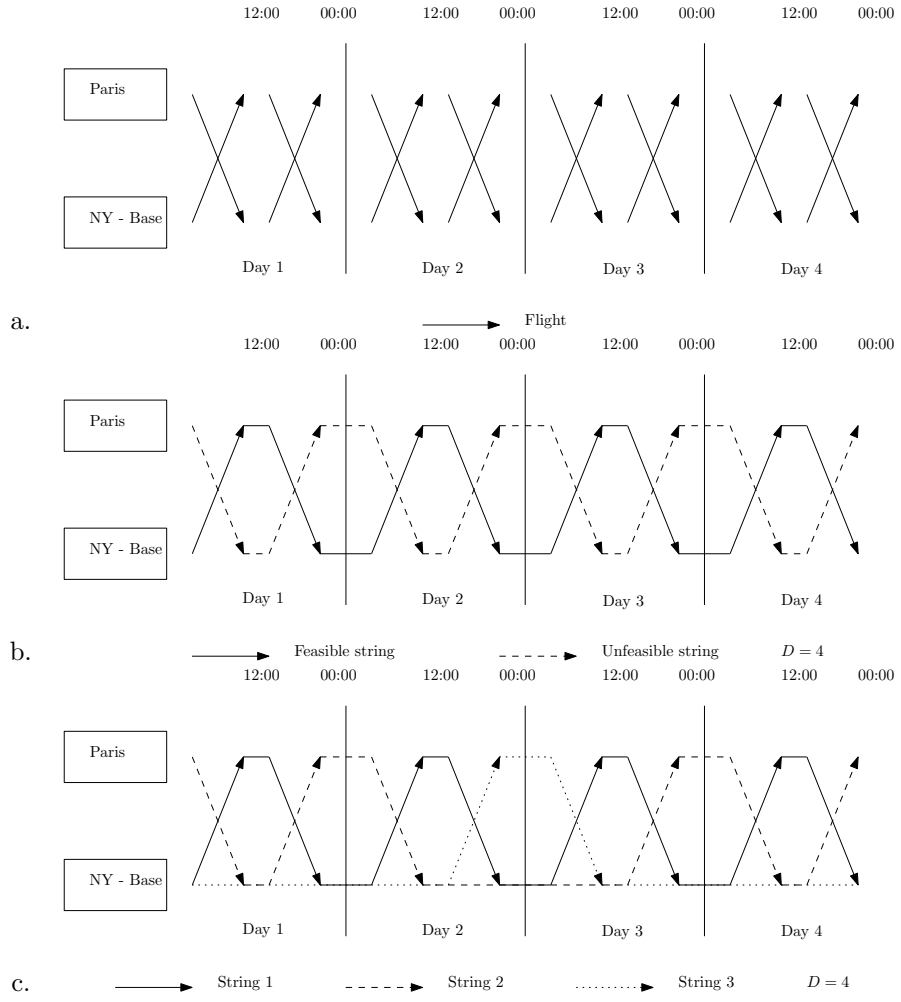

Figure 1: a. An aircraft routing instance – b. Flight strings – c. A feasible routing

f_2 is equal to the arrival airport of f_1 and $d_{f_1}^{arr} \leq d$ and $d_{f_2}^{dep} > d$. A couple of flights is a *maintenance couple* $(f_1, f_2) \in \Pi_d \subseteq \Lambda_d$ if it is a night couple $(f_1, f_2) \in \Lambda_d$ in a base $\alpha_{f_1}^{arr} \in \mathcal{B}$ such that the time interval between the two flights is sufficient to perform a maintenance check $(t_{f_2}^{dep} + \tau) - t_{f_1}^{arr} \geq \tau_M$, where τ_M is the time needed to perform a maintenance check. An example of aircraft routing instance is on Figure 2.1.

A *flights string* σ is a list of flights $\sigma = (f_1, f_2, \cdots, f_p)$ such that $\alpha_{f_i}^{arr} = \alpha_{f_{i+1}}^{dep}$ and $t_{f_i}^{arr} \leq t_{f_{i+1}}^{dep}$ for $i = 1, \ldots, p-1$. A string operates a maintenance on night d if there exist two successive flights f_i, f_{i+1} of σ such that $(f_i, f_{i+1}) \in \Pi_d$ is a maintenance couple for night d. A flight string is *feasible* if it visits a maintenance night $(f_i, f_{i+1}) \in \pi_d$ at least once in each sequence of D successive nights. A flight f is *covered* by a flight string σ if it is one of the flights of σ. Considering a flight f, the number of *days of operations* $o_f = d_f^{dep} - d$ of f with respect to σ is equal to the number of days between the departure of the flight d_f^{dep} and the last maintenance night set Π_d covered by σ. Example of flight strings are given on Figure 2.1.

The goal of the aircraft routing problem is to cover all flights with feasible flight strings. But considering one specific flight string σ, the last flight f of σ does not necessarily arrive in a maintenance base, and thus its number of days of operations o_f must be taken into account as an initial condition in the aircraft routing problem for the next period. Thus, in the aircraft routing problem, the initial positions of airplanes are given: for each day of operation $d \in [D]$ and airport α, there are initially κ_α^d airplanes that must visit a maintenance base after at most $D-d+1$ days. For each airport α and day of operation d, the final conditions are given by the γ_α^o: there are at least $\sum_{o=1}^{d} \gamma_\alpha^o$ airplanes that have had a maintenance in the last d nights. The *fleet size* is $k = \sum_{\alpha \in \mathcal{A}} \sum_{d \in [D]} \kappa_\alpha^d$.

A *routing* $\mathcal{S}$ is a collection of flight strings $\sigma \in \mathcal{S}$. A routing is *feasible* if each flights string $\sigma \in \mathcal{S}$ is feasible, each flight $f \in F$ is covered by one unique flight string $\sigma \in \mathcal{S}$, at least $\sum_{o=d}^{D} \kappa_\alpha^o$ flight strings starting in airport a visit a maintenance night after at most $D - d + 1$ days for all $d \in [D]$, and at least $\sum_{o=1}^{d} \gamma_\alpha^o$ flights string ending in airport a have visited a maintenance night in the last d days for all $d \in [D]$. A feasible routing is on Figure 2.1.

Problem Aircraft routing feasibility problem
INSTANCE: An horizon H, a time discretization T, a set of airports $a \in \mathcal{A}$, a set of bases $\mathcal{B} \subseteq \mathcal{A}$, a set of flights $F \subseteq (\mathcal{A} \times T \times D)^2$, a maximum number of days between two maintenance nights D, and for each airport $a \in A$ and day $o \in [D]$ the initial and final number of airplanes κ_α^o and γ_α^o.
QUESTION: Does a feasible routing $\mathcal{S}$ exist?

In this section, the aircraft routing feasibility problem is proved to be NP-complete in the general case, but polynomial when the number of airplanes k in the fleet is fixed, and an algorithm in $O(|F| \cdot D^k)$ is given. The remainder of the section is organized as follows: aircraft routing problem equivalence with a graph problem is proved in Section 2.2, the polynomial algorithm when k is fixed is given in Section 3.2, and the NP-completeness in the general case is proved in Section 4.

2.2 Equivalence with a graph problem

In order to solve the aircraft routing problem, we will prove the equivalence between the aircraft routing problem and a graph problem. This graph problem is formulated on graphs satisfying several specific properties. Therefore, the special-purpose notions of *equigraph* and *path partitions* are introduced.

Let $G = (V, A)$ be an acyclic directed graph. Let $\delta^-(v)$ (resp. $\delta^+(v)$) be the set of incoming (resp. outgoing) arcs from v. Let $d^-(v) = |\delta^-(v)|$ (resp. $d^+(v) = |\delta^+(v)|$) be the incoming (resp. outgoing) degree of v. A vertex $v \in V$ is a *source* if $\delta^-(v) = \emptyset$, and it is a *sink* if $\delta^+(v) = \emptyset$. We denote by S the set of sources, and by T the set of sinks. Let $I = V \backslash (S \cup T)$ be the set of *internal vertices*. A path $P \in G$ is a *source to sink path*, or $S - T$ path if it starts in a source and ends in a sink. An arc $a \in A$ is *covered* by a path P if $a \in P$. A collection of path $\mathcal{P}$ is a *path partition* of G if for each arc $a \in A$, there exists one and only one path $P \in \mathcal{P}$ such that $a \in P$. An acyclic directed graph G is an *equigraph* if each internal vertex $v \in I$ satisfies $d^-(v) = d^+(v)$.

The definition of the equigraph problem and its equivalence to the aircraft problem is done in Section 2.2.1. Arcs in the equigraph corresponds to flights or ground connections between two flights. Day separation by nights is modeled thanks to directed cut in equigraphs. General properties of equigraphs that gives polynomial time algorithms on simple instances are studied in Section 2.2.2.

2.2.1 Equivalence of the two problems

In a directed graph, a *directed cut* is an arc set C such that $C = \delta^-(U)$ for a vertex set U satisfying $\delta^+(U) = \emptyset$. Given an equigraph $G = (V, A)$, a collection of directed cuts $N_d = \delta^-(U_d)$ for $d \in [H]$ is a collection of *nights* if $U_{d+1} \subseteq U_d$ for all $d \in [H-1]$ and $U_H = T$ is the set of sinks. Such a collection is illustrated on Figure 2. The *horizon* is the number of nights H. The day of a vertex $day(v)$ is the index of the first night after v: $d = day(v)$ if $v \in U_{d-1} \backslash U_d$. Besides, the maintenance checks operated in bases during nights are modeled thanks to a given collection M_d of *maintenance arc sets*: subset M_d of N_d identifies the maintenance night arcs among the night arcs of N_d. A path P intersects a set N if $P \cap N \neq \emptyset$.

Let D be an integer called the *maintenance requirement*. A path P from sources S to sinks T necessarily intersects all nights N_d. It is a *feasible path* if it intersects a maintenance night arc at least every D nights: $P \cap \left(\cup_{o=d}^{d+D-1} M^o\right) \neq \emptyset$ for $d \in [H - D + 1]$. A path P *covers* an arc a if $a \in P$. For an arc a covered by a path P, the number of days of operations $o_P(a)$ is equal to the number of days between a and the last maintenance night M_d intersected. Thus, a path is feasible if the number of days of operations of all its arcs is smaller than the maintenance constraint: $o_P(a) \leq D$ for all $a \in P$.

Initial and final constraints are given on the number of days of operations at the beginning (source) and at the end (sink) of each feasible path. For each source (resp. sink) s and integer $o \in [D]$, let κ_s^o (γ_s^o) be these constraints. A collection of *feasible* paths $\mathcal{P}$ is a *feasible routing* if the following conditions are satisfied: first, each flight is covered exactly once. Second, there are at least $\sum_{o=d}^{D} \kappa_s^o$ paths starting in source s that visits one of the first $D - d + 1$ maintenance night arc sets $\bigcup_{o=1}^{D-d+1} M_o$ for all $d \in [D]$. Finally there are at least

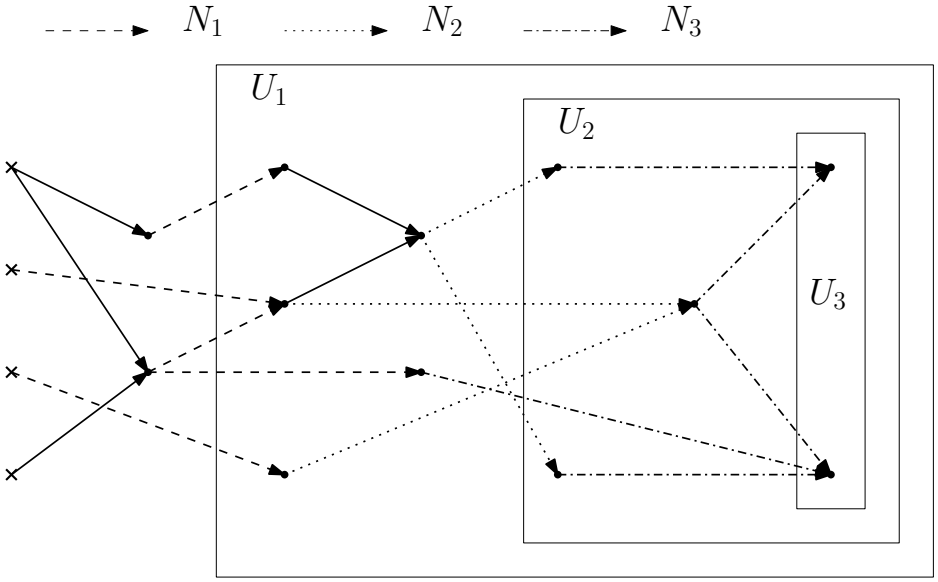

Figure 2: A set of nights on an equigraph

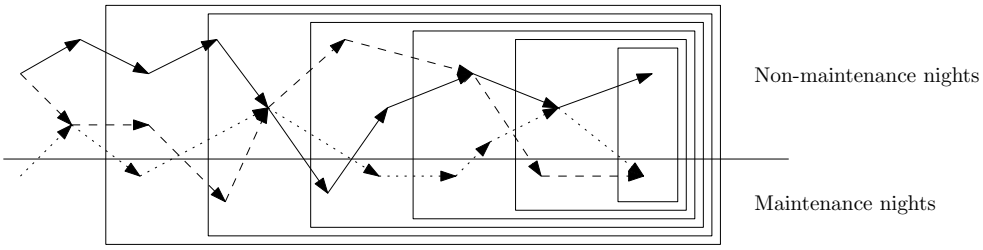

Figure 3: A feasible routing

$\sum_{o=1}^{d} \gamma_t^o$ paths ending in sink t that visits one of the last d maintenance night arc sets $\bigcup_{o=H-d+1}^{H} M_o$ for all $d \in [D]$. Integer κ_s^o are the initial requirements, and γ_t^o are the final requirements. A feasible routing is plotted on Figure 3.

The equigraph routing problem can be stated as follows:

Problem Equigraph routing problem
INSTANCE: An equigraph $G = (V, A)$, a collection of nights N_d and maintenance nights $M_d \subseteq N_d$, a maintenance requirement D, initial requirements κ_s^o and final requirements γ_s^o.
QUESTION: Does a feasible routing exist?

Theorem 1. *Aircraft routing problem and equigraph routing problem are equivalent: each problem can be reduced to the other one in linear time.*

Thanks to this theorem, algorithm and complexity for the equigraph routing problem and the aircraft routing problem are equivalent. The remaining of this

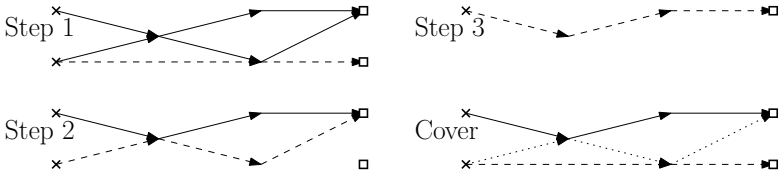

Figure 4: Greedy algorithm to find a path partition on an equigraph

paper focus on equigraph routing problem. In the following long proof, only the methods used to build the equivalent instance are practically interesting.

Proof. The proof of Theorem 1 is straightforward. A detailed version is given in appendix. When reducing aircraft routing to equigraph routing, the natural idea is to use airport-time couples as vertices and flights as arcs. The only technical issue is on the number of ground arcs relying vertices representing the same airport at different time. It must be chosen in order to obtain an equigraph. Counting the number of arrivals minus the number of departures at an airport across gives a solution to this point. Conversely, when reducing equigraph routing to aircraft routing, a single airport is added for each vertex. The only technical point is to build a relevant time index, which can be done thanks to a depth first exploration of the equigraph. □

2.2.2 $S-T$ paths partitions of graphs

In this section, properties on equigraphs and simple path partition algorithms on equigraph are given.

Proposition 2. *An acyclic directed graph G admits a $S-T$ path partition if and only if it is an equigraph.*

Lemma 3. *If P is a source to sink path in an equigraph G, then $G\backslash P$, the graph obtained by removing the arcs of P and the newly isolated vertices is an equigraph.*

Proof. No incoming (resp. outgoing) arcs are added to sources (resp. sinks), and if v is an internal vertex on P, both its incoming and outgoing degree are decreased by the same quantity. □

This lemma can be seen as a linear time *greedy algorithm to find a path partition on an equigraph*. Indeed, as for each internal vertex, $|\delta^-(v)| = |\delta^+(v)|$, a path from sources to sinks is always found by picking each nodes successor. Thus, a path partition is found by picking a path from sources to sink and remove it from the equigraph, as shown in Figure 4. At each step, the graph obtained by removing a path is still an equigraph whose number of sources or sinks has been decreased by one. Because it is acyclic, an equigraph without sources and sinks is empty, and the algorithm finally gives a path partition of the equigraph.

Proof of Proposition 2. First *prove that the existence of a path partition of G implies that G is an equigraph.* Let $\mathcal{P}$ be a path partition of G, let v be an internal vertex of G, and d be the number of paths in $\mathcal{P}$ that cover v. Then, as

each path covering v enters in v by an arc in $\delta^-(v)$ and exits by an arc in $\delta^+(v)$, and paths in $\mathcal{P}$ are arc disjoint, we necessarily have $d = |\delta^-(v)| = |\delta^+(v)|$.

Conversely, the greedy algorithm deduce from Lemma 3 gives the existence of a path partition for any equigraph G. □

Summing $d^+(v) = d^-(v)$ for all internal vertices $v \in I$ gives the equality of incoming and outgoing degrees for the internal vertices $d^-(I) = d^+(I)$. Thus, we obviously have $d^+(S) = d^-(T)$ and this number is equal to the number of path in a path partition of G. Let $d^+(S)$ be the *flow* of equigraph G.

Remark 1. Let P be a collection of $d^+(S) = d^-(T)$ disjoint paths in an equigraph. Then these paths cover all the arcs of the graph. Indeed, removing all these paths from the graph gives an equigraph without sources and sinks, which is thus necessarily empty.

Finally, the following result is a characterization of equigraphs thanks to directed cuts. A directed cut $\delta^-(U)$ is *terminal* if $T \subseteq U$.

Proposition 4. *An acyclic graph G is an equigraph if and only if all terminal directed cuts $C = \delta^-(U)$ have the same cardinality $|C| = d^+(S) = d^-(T)$.*

Proof. Let G be an equigraph, C be a terminal directed cut, and $\mathcal{P}$ be a path partition of G, then each path $P \in \mathcal{P}$ intersects C exactly once: there is a one to one mapping from $\mathcal{P}$ to C, which gives the "only if" direction.

Conversely, let G be a graph that is not an equigraph. There exist a vertex v such that $d^+(v), d^-(v) > 0$ and $d^+(v) \neq d^-(v)$. Suppose first that $|\delta^-(v)| < |\delta^+(v)|$. Let U be the union of the set of vertices u such that there is a path from v to u with the set of terminals. Then $\delta^-(U)$ and $\delta^-(U \cup \{v\})$ are two terminal directed cuts, and $|\delta^-(U \cup \{v\})| \neq |\delta^-(U)|$. The case with $|\delta^-(v)| > |\delta^+(v)|$ is analogous. □

3 Network state graphs and integer linear program for aircraft routing

Equigraph routing problem is a path partition problem on an equigraph with additional constraints. In this section, we introduce the notion of *state graph* which enables to enforce constraints on paths in a path partition, and apply it to aircraft routing. This notion of state graphs enables us to prove that aircraft routing is polynomial at number of airplanes fixed, and to introduce a new integer linear formulation for aircraft routing.

3.1 Network-state graph formalism and aircraft routing

In the equigraph routing problem, the constraint enforced on paths to obtain a feasible path partition is to visit a maintenance night set every D days. A control must be kept at all time on the number of days since the last visit of each aircraft in a maintenance base. The idea behind network state formalism is to define this number of days since the last visit in a maintenance station as the state of the flight, and to identify the state of each flight instead of the aircraft that will cover it.

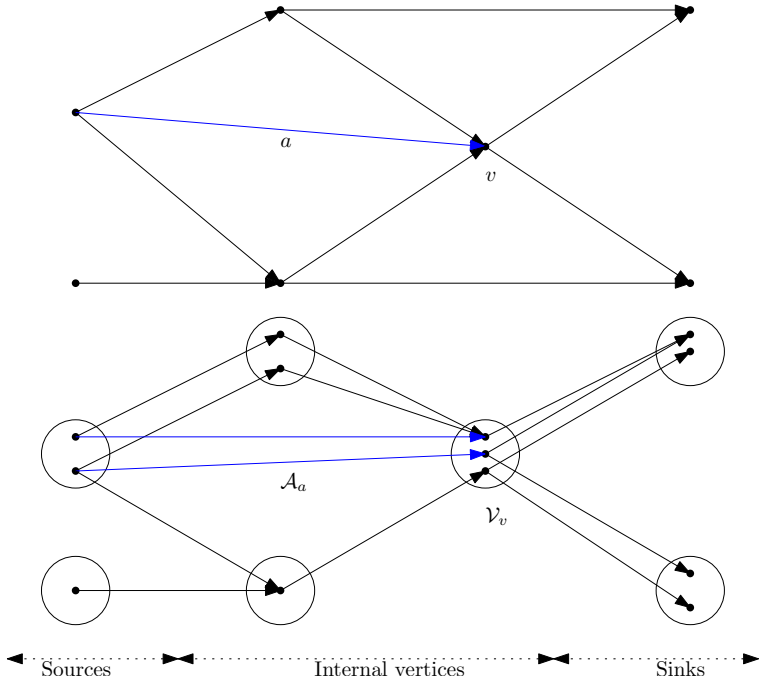

Figure 5: A network graph and its state graph

Let $G = (V, A)$ be an acyclic directed graph called the *network graph*. Vertex set V is equal to $S \cup I \cup T$ where S is the set of sources, I is the set of internal vertices, and T is the set of sinks. An acyclic directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ is a *state graph* on G if it can be described as follows. To each vertex $v \in V$ corresponds a *state set* $\mathcal{V}_v$. Vertex v is the *network vertex* of each state vertex in $\mathcal{V}_v$. The set of vertices of state graph $\mathcal{G}$ is the union of the state sets $\mathcal{V} = \bigcup_{v \in V} \mathcal{V}_v$. These vertices are called *state vertices*.

We describe now the arcs of a state graph. Given two state vertices ϑ_1 and ϑ_2 and there respective network vertices v_1 and v_2, there can be an arc $\alpha = (\vartheta_1, \vartheta_2)$ only if $a = (v_1, v_2)$ is an arc of the network graph. Arc a is the *network arc* of α, and α is a *state arc* of a. Let $\mathcal{A}_a$ be the set of state arcs of a. Let v be a network vertex and $a \in \delta^+(v)$ be a network arc. There is at most one arc α in $\mathcal{A}_a$ outgoing from each state vertex $\vartheta \in \mathcal{V}_v$. The set of arcs $\mathcal{A}$ of the state graph is the union of the set of states arcs $\mathcal{A} = \bigcup_{a \in A} \mathcal{A}_a$. The set of sources $\mathcal{S}$, internal vertices $\mathcal{I}$ and sinks $\mathcal{S}$ of state graph $\mathcal{G}$ are defined as the union of the states sets of the sources, internal vertices, and sinks in the network graph: $\mathcal{S} = \bigcup_{s \in S} \mathcal{V}_s$, $\mathcal{I} = \bigcup_{v \in I} \mathcal{V}_i$, and $\mathcal{T} = \bigcup_{t \in T} \mathcal{V}_t$. Each arc α or vertex $\vartheta \in \mathcal{I}$ such that there is not both a path from sources to it and a path from it to the sinks is removed from the state graph. Thus, the definition of sources , sinks, and internal vertices are consistent. An example of network graph and and state graph is on Figure 5.

Given an instance $(G, N_d, M_d, D, \kappa_s^o, \gamma_s^o)$ of the equigraph routing problem, its corresponding *routing state graph* is the state graph on $G = (V, A)$ built as follows: for each vertex $v \in V$, build a set $\mathcal{V}_s = \{\vartheta_v^1, \ldots, \vartheta_v^D\}$ of D state vertices. We define the state arcs in order to ensure that *for each source to sink path π in*

$\mathcal{G}$ *covering* ϑ_v^o*, the number of days since the last visit of* $P(\pi)$ *in a maintenance arc is equal to* o. Therefore, for each arc $a = (v_1, v_2) \in A \setminus (\cup_d N_d)$ and $o \in [D]$, add to $\mathcal{A}_a$ the state arc $(\vartheta_{v_1}^o, \vartheta_{v_2}^o)$. State does not change on a non-night arc. For each arc $a = (v_1, v_2) \in \cup_d M_d$ and $o \in [D]$, add the arc $(\vartheta_{v_1}^o, \vartheta_{v_2}^1)$ to $\mathcal{A}_a$. After a maintenance arc, the number of day since the last maintenance night is equal to 1. Finally, for each arc $a = (v_1, v_2) \in (\cup_d N_d \setminus \cup_d M_d)$ and $o \in [D-1]$, add the arc $(\vartheta_{v_1}^o, \vartheta_{v_2}^{o+1})$ to $\mathcal{A}_a$. After a non-maintenance night, the number of days since the last maintenance night has increased by one. If D days have elapsed since the last visit of an airplane in v_1 to a maintenance night arc, then it cannot take a non-maintenance night arc.

For each source to sink path $\pi \in \mathcal{G}$, there is a unique source to sink path $P(\pi) \in G$. We have the the following lemma.

Lemma 5. *Let* $\mathcal{G}$ *be a routing state graph on an instance* $G, N_d, M_d, \kappa_s^o, \gamma_s^o$ *of equigraph routing problem. For each* $S-T$ *path* $P \in G$*, there exist a* $S-T$ *path in* $\pi \in \mathcal{G}$ *if and only if* P *is a feasible source to sink path in* G*.*

Proof. Let $a = (v_1, v_2)$ be an arc in $P(\pi)$ and $o_P(a)$ the number of days since the last visit of P in a maintenance night arc, then the unique state arc α in $\mathcal{A}_a \cap \pi$ starts in $\vartheta_v^{o_P(a)}$, and thus $o_P(a) \leq D$. □

Finally, initial and final conditions can be enforced on a state graph as follows: for each state source ϑ, define κ_ϑ as the number of paths starting in ϑ, and for each state sink ϑ, define κ_ϑ as the number of paths ending in ϑ. A *network path partition* is a collection Π of source to sink paths π in $\mathcal{G}$ such that one and exactly one state arc α in each state arc set $\mathcal{A}_a$ is covered by a path $\pi \in \Pi$, exactly κ_ϑ paths start in source ϑ, and κ_ϑ paths end in sink ϑ. By taking the network arc a of each state arc α in a path π, path π induces a path partition P_π in G, and thus Π induces a path partition $\mathcal{P}_\Pi = \bigcup_{\pi \in \Pi} P_\pi$ in G.

In a routing state graph, for each source s and state source ϑ_s^o, we define $\kappa_{\vartheta_s^o} = \kappa_s^o$, and for each sink s and state sink ϑ_s^o, we define $\kappa_{\vartheta_s^o} = \kappa_s^o$.

Lemma 6. *Let* $\mathcal{G}$ *be a routing state graph on an instance* $(G, N_d, M_d, \kappa_s^o, \gamma_s^o)$ *of equigraph routing problem. A feasible routing corresponds to a unique network path partition in* $\mathcal{G}$*.*

Proof. Let Π be a network path partition. Due to Lemma 5, each path in $\mathcal{P}(\Pi)$ is a feasible equigraph routing path. The uniqueness of Π comes from the fact that the number of arcs in a state arc set $\mathcal{A}_a$ outgoing from a state vertex ϑ is at most one. □

Thus, equigraph routing problem can linearly be reduced to the following problem.

Problem Network path partition
INSTANCE: A network graph G, and its state graph $\mathcal{G}$.
QUESTION: Does a network path partition exist?

A *state equigraph* is a partial graph of $\mathcal{G}$ which is an equigraph and such that for each arc $a \in A$, $|\mathcal{G}_\Pi \cap \mathcal{A}_a| = 1$. *If* Π *is a network path partition in* $\mathcal{G}$*, then* $\mathcal{G}_\Pi = \cup_{\pi \in \Pi} \pi$ *is state equigraph.* The following Lemma is the root of the solution methods to find a network path partition.

Lemma 7. *A state graph admits a network path partition if and only if it admits a state equigraph. Besides, a network path partition can be obtained from a state equigraph in linear time.*

Proof. Let Π be a network path partition in $\mathcal{G}$, then $\mathcal{G}_\Pi = \cup_{\pi\in\Pi}\pi$ is a state equigraph. Reciprocally, let $\mathcal{H}$ be a state equigraph, then by Lemma 3, it can be partitioned in linear time in source to sink paths in linear time. □

In Section 3.2, an algorithm for network path partition problem is given: it is polynomial for fixed value of the flow of the network graph. It induces a polynomial algorithm at number of airplanes fixed for the aircraft routing problem. Finally, in Section 3.3, a compact linear program to solve the network path partition problem and consequently the aircraft routing problem is introduced.

3.2 Polynomial algorithm for network paths partition problem

This section is devoted to the proof of Theorem 8.

Theorem 8. *The aircraft routing problem is polynomial for fixed number of airplanes k. It can be solved in time bounded from above by $B_0 = 2nD^k$, where k is the number of airplanes and n the number of flights.*

Proof of theorem 8 consists in a polynomial algorithm inspired of the pebbling game algorithm to solve the integer multicommodity flow problem [17]. In the remaining of the section, three lemmas are introduced to be able to prove Theorem 8 at the end of the section.

A complete ordering $v_1, v_2, \ldots, v_n$ on the vertices of an acyclic directed graph is a *topological ordering* if $(v_i, v_j) \in A$ implies $i < j$. A breadth first search gives such an ordering. Given a topological ordering $v_1, v_2, \ldots, v_n$ of the vertices of an acyclic directed graph, define U_i as the set of vertices strictly after v_i in the ordering, $U_i = \{v_j \in V | j > i\}$, and $U_0 = V$. As $v_1, v_2, \ldots, v_n$ is a topological ordering, $C_i = \delta^-(U_i)$ is a terminal directed cut. This way, to each topological ordering $v_1, v_2, \ldots, v_n$ corresponds a collection of directed cuts $C_0, C_1, \ldots, C_n$ with $C_i = \delta^-(U_i)$ and $V = U_0 \subsetneq U_1 \subsetneq \cdots \subseteq U_n = \emptyset$. A topological ordering and its directed cuts collection are illustrated on Figure 6. Besides, we have:

$$\begin{aligned} U_{i-1} &= \{v_i\} \cup U_i \\ C_{i-1} \backslash C_i &= \delta^-(v_i) \\ C_i \backslash C_{i-1} &= \delta^+(v_i) \end{aligned}$$

Finally, define the partial graph G_i as the union $\bigcup_{j=0}^{i} C_i$ of the directed cuts C_j whose index j is smaller than i. As all the terminal directed cuts of G_i are terminal directed cuts of G, they have all the same cardinality and proposition 4 ensures that G_i is an equigraph. Let $\mathcal{G}_i$ be the partial graph of $\mathcal{G}$ whose arcs are in $\mathcal{A}^i = \cup_{a\in G_i} A_a$. Let T_i be the set of sinks of G_i, and $\mathcal{T}_i$ be the set of sinks of $\mathcal{G}_i$.

A *distribution of pebble* on $\mathcal{T}_i$ is an application $X_i : \mathcal{T}_i \to \mathbb{Z}^+$; it is *reachable* if there exist a network path partition Π_i on $\mathcal{G}_i$ such that there are $X_i(\vartheta)$ paths π in Π ending in ϑ for each state vertex $\vartheta \in \mathcal{T}_i$. Partition Π_i is *ending in* X_i. Let

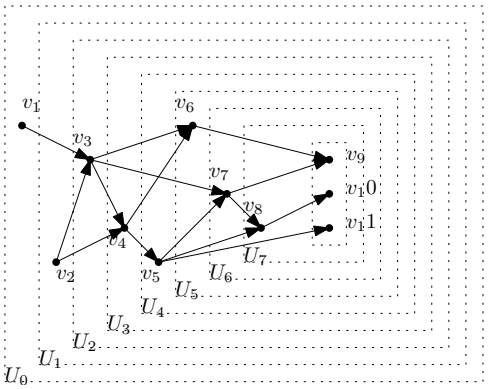

Figure 6: A topological ordering and its ordering cuts collection

χ_i be the *set of reachable distributions on* $\mathcal{T}_i$. Let X_i be a reachable distribution in χ_i, and Π_i a corresponding path partition of $\mathcal{G}_i$. A distribution X_{i+1} on $\mathcal{T}_{i+1}$ can be *reached from* X_i if any network path partition Π_i of $\mathcal{G}_i$ ending in X_i can be completed in a network path partition Π_{i+1} on $\mathcal{G}_{i+1}$ ending in X_{i+1}. The idea of the algorithm is to deduce χ_{i+1} from χ_i as the union of the distributions X_{i+1} that can be reached from a distribution $x_i \in \chi_i$ thanks to a legal move of pebbles.

Lemma 9. *Pebble distribution X_{i+1} on $\mathcal{T}_{i+1}$ is reachable from distribution X_i on $\mathcal{T}_i$ if and only if the following conditions are satisfied:*

1. *Pebbles that are not in $\mathcal{V}_{v_i}$ are unmoved.*
2. *A pebble initially on vertex ϑ_1 can be moved to vertex ϑ_2 if and only if $(\vartheta_1, \vartheta_2)$ is an arc in $\mathcal{G}$.*
3. *Only one pebble goes through each arc set $\mathcal{A}_a$ with $a \in \delta^+(v)$.*

A pebble move satisfying these conditions is called a legal move. Let $m(\vartheta)$ be the number of pebbles moved to ϑ, then $X_{i+1}(\vartheta) = X_i(\vartheta) + m(\vartheta)$ if $\vartheta \in \mathcal{T}_i \cap \mathcal{T}_{i+1}$ and $X_{i+1}(\vartheta) = m(\vartheta)$ otherwise.

Proof. Suppose that a legal move lead from X_i to X_{i+1}. Let Π_i be a network path partition ending in X_i, affect each pebble moved from $\vartheta \in \mathcal{T}_i \backslash \mathcal{T}_{i+1}$ to a path $\pi \in \Pi_i$ ending in ϑ, and complete π by the arc α crossed by its pebble. The completed paths form a network path partition on $\mathcal{G}_{i+1}$ ending in $\mathcal{T}_{i+1}$.

Conversely, if X_{i+1} is reached from X_i in the network path partition Π_{i+1}, then move one pebble along each arc of $\Pi_{i+1} \cap (\cup_{a \in C_i} \mathcal{A}_a)$ to obtain a legal move. □

Define $\phi_i : \chi_i \to \wp(\chi_{i+1})$ as the operator that associates to a reachable distribution X_i the set of distributions X_{i+1} reachable from X_i. Lemma 9 ensures that $\chi_{i+1} = \cup_{X_i \in \chi_i} \phi_i(\chi_i)$. Define $\chi = \cup_i \chi_i$ as the set of reachable distributions, define *sources distributions* $X \in \mathcal{I} \subseteq \chi$ as distributions of pebbles on the sources, and *sinks distributions* $X \in \mathcal{J} \subseteq \chi$ are distributions on the sinks. The acyclic directed *distribution graph* associated to the network state graph is defined as follows: one vertex for each reachable pebble distribution

$X \in \chi$, and for each pebble distribution X in χ, one arc between X and each distribution X' in $\Phi(X)$.

Lemma 10. *There exist a network path partition of $\mathcal{G}$ if and only if there exist a path from a source distribution to a sink distribution in the distributions graph.*

Proof. A path from a source distribution to a sink distribution is obtained from a network path partition by a applying recursively Lemma 9.

Conversely, given a path $\mathcal{D}$ from a source distribution to a sink distribution, let $\mathcal{G}'$ be the partial graph of $\mathcal{G}$ obtained by taking the arcs crossed by a pebble during one of the legal move of $\mathcal{D}$. This graph is a state equigraph, and thus a path partition can be deduced from it in linear time by applying recursively Lemma 3. □

Finally, a last lemma on the cardinality of the arc set of the distribution graph is needed to be able to prove Theorem 8.

Lemma 11. *The number of arcs in the distribution graph of a state graph $\mathcal{G}$ is bounded from above by $B_0 = |V|(\max_{v\in V} |\mathcal{V}_v|)^k$, where k is the flow of equigraph G.*

A tighter bound $B_1 \leq B_0$ is

$$B_1 = \sum_{v \in S \cup I} \left(\prod_{u \in S_v \cap S_{v+1}} \begin{pmatrix} p_v(u) + |\mathcal{V}_u| - 1 \\ |\mathcal{V}_u| - 1 \end{pmatrix} \right) \cdot |\mathcal{V}_v|^{d(v)} \tag{1}$$

where $p_v(u) = |C_v \cap \delta^-(u)|$ is the number of pebbles on u when v is played.

Proof. As arcs in the distribution graphs are between χ_i and χ_{i+1}, it suffices to bound the number of legal move between χ_i and χ_{i+1}. As each network arc $a \in A$ is crossed exactly once, the number of pebbles on vertices in $\mathcal{V}_u$ is identical for each distribution in χ_v. Define $p_v(u)$ as the number of pebbles on state vertices in $\mathcal{V}_u$ in distribution of χ_v. Thus, the number of legal moves is equal to the number of distributions of pebbles on $S_v \cap S_{v+1}$, which corresponds to the pebbles which do not move, multiplied by the number of legal moves of the moving pebbles.

For each vertex u in $S_v \cap S_{v+1}$, the number of way to distribute $p_v(u)$ pebbles on $|\mathcal{S}_u|$ states is equal to $\begin{pmatrix} p_v(u) + |\mathcal{V}_u| - 1 \\ |\mathcal{V}_u| - 1 \end{pmatrix}$.

Let $\vartheta_1, \vartheta_2, \ldots, \vartheta_\ell$ be the state vertices in $\mathcal{V}_v$ and x_i be the number of pebbles on ϑ_i in distribution X_i. Then $x_1 + x_2 + \cdots + x_\ell = d(v)$. As, first, in the path partition, each arc of G is covered exactly once and, second, for each network arc, there is at most one state arc outgoing from each state vertex of $\mathcal{V}_v$, the number of legal moves is bounded from above by the number of partition of the $d(v)$ arcs in $\delta^+(v)$ in sets of cardinal $x_1, x_2, \ldots, x_i$ (which corresponds to the origin of the state arcs): $\begin{pmatrix} d(v) \\ x_1, x_2, \ldots, x_m \end{pmatrix}$. Thus, the total number of legal move is bounded from above by

$$\begin{aligned} & \left(\prod_{u \in S_v \cap S_{v+1}} \begin{pmatrix} p_v(u) + |\mathcal{V}_u| - 1 \\ |\mathcal{V}_u| - 1 \end{pmatrix} \right) \cdot \sum_{x_1 + x_2 + \ldots + x_l = d(v)} \begin{pmatrix} d(v) \\ x_1, x_2, \ldots, x_\ell \end{pmatrix} \\ = & \left(\prod_{u \in S_v \cap S_{v+1}} \begin{pmatrix} p_v(u) + |\mathcal{V}_u| - 1 \\ |\mathcal{V}_u| - 1 \end{pmatrix} \right) \cdot l^{d(v)} \end{aligned} \tag{2}$$

Summing the upper bound $|\Phi(\chi_i)|$ gives the upper bound B_1 on $|\Phi(\chi)|$. $B_1 \leq B_0$ is obtained from $\begin{pmatrix} p_v(u) + |\mathcal{V}_u| - 1 \\ |\mathcal{V}_u| - 1 \end{pmatrix} \leq |\mathcal{V}_u|^{p_v(u)}$ and $\sum_{u \in S_v} p_v(u) = k$. □

Proof of Theorem 8. Proof of Theorem 1 ensures that an aircraft routing problem instance $(H, T, \mathcal{A}, \mathcal{B}, F, D, \kappa_\alpha^0, \kappa_\beta^0)$ can be reduced to an equigraph routing problem instance $(G, N_d, M_d, \kappa_s^o, \kappa_t^o)$ whose number of arcs is bounded form above by $2n$ where n us the number of flights. Lemma 6 ensures that this equigraph routing problem can be reduced to a network path partition problem on a state graph $\mathcal{G}$ on G such that for each vertex v of G, state vertices set $\mathcal{V}_v$ has a cardinality no greater than $\mathcal{D}$. Due to Lemma 7, the existence of a network state path partition is equivalent to the existence of a state equigraph. Finally, Lemma 10 ensures that the existence of state equigraph is equivalent to the existence of a path from a source distribution to a sink distribution in $\mathcal{G}$.

As the distribution graph of a network state graph is acyclic, the complexity of a path finding algorithm in this graph is linear in the number of arcs in the distributions graph. Finally Lemma 11 ensures that the number of arcs in the distribution graph is bounded from above by $|V|(\max_{v \in V} |\mathcal{V}_v|)^k$, which gives the result because $\max_{v \in V} |\mathcal{V}_v| \leq D$ and $|V| \leq 2n$. □

.

Remark 2. In the proof to Theorem 8, we have proved that network path partition problem can be solved in polynomial time when the flow of the network equigraph is fixed.

Remark 3. A shortest path algorithm in distribution graph solves the network path partition optimization problem introduced in the next section. Its complexity admits the same bound B_0 and B_1 as the path finding algorithm.

3.3 Linear program for aircraft routing problem

In this section, we introduce a compact linear program to solve efficiently the network path partition problem introduced in Section 3.1. A special case of this program gives a compact linear program to solve aircraft routing problem.

Let $\mathcal{G}$ be a state graph on equigraph G, and for each α in $\mathcal{A}$, let $c_\alpha \in \mathbb{R}$ be a cost attached to state arc α. In the case of aircraft routing, cost c_α could represent the risk of operating a flight after o days without maintenance. The *cost of a network path partition* Π is defined as the sum of the cost of state arcs in the partition $C(\Pi) = \sum_{\alpha \in \pi \in \Pi} c_\alpha$. The network path partition optimization problem can be stated as follows:

Problem Network path partition optimization problem
INSTANCE: A network graph G, its state graph $\mathcal{G}$, and costs c_α
SOLUTION: A minimum cost network path partition

To solve the network path partition optimization problem, we introduce the

following linear integer program:

$$\begin{array}{lll} \min & \sum_{\alpha\in\mathcal{A}} c_\alpha x_\alpha & \\ s.t. & \sum_{\alpha\in\delta^-(\vartheta)} x_\alpha = \sum_{\alpha\in\delta^+(\vartheta)} x_\alpha & \forall\vartheta\in\mathcal{A} \\ & \sum_{\alpha\in\mathcal{A}_a} x_\alpha = 1 & \forall a\in A \\ & \sum_{\alpha\in\delta^+(\vartheta)} x_\alpha = s_\vartheta & \forall\vartheta\in\mathcal{S} \\ & \sum_{\alpha\in\delta^-(\vartheta)} x_\alpha = t_\vartheta & \forall\vartheta\in\mathcal{T} \\ & x_\alpha\in\{0,1\} & \forall\alpha\in\mathcal{A} \end{array} \tag{3}$$

The first constraint in (3) ensures that the partial graph $(\mathcal{V},\mathcal{B})$ where $\mathcal{B}=\{\alpha|x_\alpha=1\}$ is an equigraph, and the second constraint ensures that exactly one arc $\alpha\in\mathcal{A}_a$ belongs to $\mathcal{B}$ for each arc $a\in A$. As a consequence, $(\mathcal{V},\mathcal{B})$ is a state equigraph. Besides, if two network path partitions shares Π_1 and Π_2 sharing the same state equigraph, then $\bigcup_{\alpha\in\pi\in\Pi_1}\alpha=\bigcup_{\alpha\in\pi\in\Pi_2}\alpha$ and therefore $C(\Pi_1)=C(\Pi_2)$. As a consequence, the cost of a network path partition only depends of its state equigraph, and therefore, an optimal solution to program (3) gives an optimal network path partition.

Remark 4. Symbol = can be changed in $\geq$ or in $\leq$ in the second constraint of Program (3) except for the sources arcs (or even for the sources arcs but adding the inequality $\sum_{\alpha\in\delta^+(\mathcal{S})} x_\alpha \leq |\delta^+(S)|$). Indeed, the flow intersecting each directed cut is equal. Thus, for each cut C, $\sum_{a\in C}\left(\sum_{\alpha\in\mathcal{A}_a} x_\alpha\right)=|C|$ and $\left(\sum_{\alpha\in\mathcal{A}_a} x_\alpha\right)\geq 1$ implies $\left(\sum_{\alpha\in\mathcal{A}_a} x_\alpha\right)=1$ for all $a\in C$.

3.3.1 Linear relaxation and column generation

The traditional linear programming approach to solve the aircraft routing problem use column generation where columns are feasible source to sink paths [5]. Column generation can be used to solve the network path partition problem, but as we prove in this section, compact linear program (3) has the same continuous relaxation as the column generation program. Using paths as columns, the master problem of the column generation approach stands as follows:

$$\begin{array}{lll} \min & \sum_{\pi\in\mathcal{F}}\left(\sum_{\alpha\in\pi} c_\alpha\right) y_\pi & \\ s.t. & \sum_{\pi \text{ covering } a} y_\pi = 1 & \forall a\in A \\ & y_\pi\in\{0,1\} & \forall\pi\in\mathcal{F} \end{array} \tag{4}$$

where $\mathcal{F}$ is the set of source to sink paths in $\mathcal{G}$.

Proposition 12. *Linear relaxation of path formulation* (4) *and linear relaxation of compact formulation* (3) *have the same optimal value.*

Proof. This proposition is a corollary of Minkowski-Weyl theorem for convex polyhedra. A constructive proof is given in appendix. □

This proposition shows that compact linear integer program (3) is an efficient substitute to the tradition column generation approach to solve the aircraft routing problem.

Remark 5. Proposition 12 is a corollary of Minkowski-Weyl theorem for convex polyhedra.

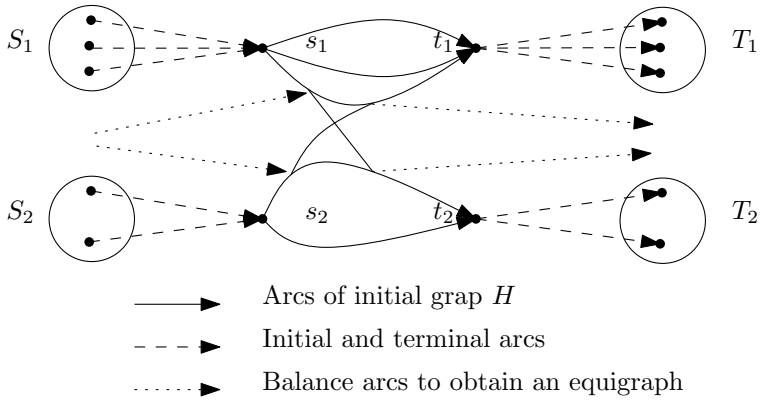

Figure 7: Equigraph G as an extension of initial graph H

4 Aircraft routing NP-completeness

The two-commodities arc-disjoint paths problem on acyclic directed graph is NP-complete [15].

Problem Two-commodities arc-disjoint paths on acyclic directed graph
INSTANCE: A directed acyclic graph $G = (V, E)$, vertices s_1 and s_2 called sources, t_1 and t_2 called terminals, two non negative R_1 and R_2
SOLUTION: R_i arc paths forms s_i to t_i for $i = 1, 2$

Theorem 13. *Aircraft routing problem is NP-complete, even restricted to two days short horizon problem*

Proof. The two-commodities arc-disjoint paths problem on acyclic directed graph can be reduced to aircraft routing

Let $H, s_1, s_2, t_1, t_2, R_1, R_2$ be an instance of the two-commodities arc-disjoint paths problem on acyclic directed graph. The graph G is extended from H in the following way: $2R_1 + 2R_2$ sources vertices are added to form S_1, t_1 and S_2, T_2, an arc is added from each vertex of S_i to s_i and from t_i to each vertex of T_i . Other vertices are internal vertices. If v is such that $\delta^-(v) > \delta^+(v)$, then $\delta^-(v) - \delta^+(v)$ sources are added with one arc between each of these and v, and $\delta^+(v) - \delta^-(v)$ terminal linked to v are added if $\delta^+(v) > \delta^-(v)$. Thus, G is an equigraph. This extension is illustrated on Figure 7.

Suppose that two-commodities arc-disjoint paths problem on acyclic directed graph admits a solution $\mathcal{P}$. Then, applying Lemma 3 for each path P of $\mathcal{P}$, $G \backslash \mathcal{P}$ is still an equigraph, and can be covered by arc disjoint paths using the greedy algorithm relying on Lemma 3 . Thus, $\mathcal{P}$ can be extended to an equigraph cover of G with R_i paths from S_i to T_i, thus it is a solution to aircraft routing problem.

Conversely, suppose that aircraft routing problem admits a solution $\mathcal{P}$. Then $\mathcal{P} \cap H$ is a solution to the two-commodities arc-disjoint paths problem. □

Research directions

Further research is currently done to use the network state formalism introduced in this section to find an aircraft minimize the expected costs of delay in the network of flights. Another axis is the use of compact linear program (3) in an integrated aircraft routing and crew pairing optimization problem.

References

[1] Jeph Abara. Applying integer linear programming to the fleet assignment problem. *Interfaces*, 19(4):20–28, 1989.

[2] Ranga Anbil, Rajan Tanga, and Ellis L. Johnson. A global approach to crew-pairing optimization. *IBM Systems Journal*, 31(1):71–78, 1992.

[3] Michael Ball and Anito Roberts. A graph partitioning approach to airline crew scheduling. *Transportation Science*, 19(2):107–126, 1985.

[4] Cynthia Barnhart, Peter Belobaba, and Amedeo R Odoni. Applications of operations research in the air transport industry. *Transportation Science*, 37(4):369, 2003.

[5] Cynthia Barnhart, Natashia L Boland, Lloyd W Clarke, Ellis L Johnson, George L Nemhauser, and Rajesh G Shenoi. Flight string models for aircraft fleeting and routing. *Transportation Science*, 32(3):208–220, 1998.

[6] Cynthia Barnhart, Timothy S Kniker, and Manoj Lohatepanont. Itinerary-based airline fleet assignment. *Transportation Science*, 36(2):199–217, 2002.

[7] John E Beasley and B Cao. A tree search algorithm for the crew scheduling problem. *European Journal of Operational Research*, 94(3):517–526, 1996.

[8] M Berge. Timetable optimization: Formulation, solution approaches, and computational issues. In *AGIFORS proceedings*, pages 341–357, 1994.

[9] Hai D Chu, Eric Gelman, and Ellis L Johnson. Solving large scale crew scheduling problems. *European Journal of Operational Research*, 97(2):260–268, 1997.

[10] Lloyd Clarke, Ellis Johnson, George Nemhauser, and Zhongxi Zhu. The aircraft rotation problem. *Annals of Operations Research*, 69:33–46, 1997.

[11] Amy Mainville Cohn and Cynthia Barnhart. Improving crew scheduling by incorporating key maintenance routing decisions. *Operations Research*, 51(3):387–396, 2003.

[12] Jean-François Cordeau, Goran Stojković, François Soumis, and Jacques Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation science*, 35(4):375–388, 2001.

[13] Guy Desaulniers, J Desrosiers, Y Dumas, S Marc, B Rioux, Marius M Solomon, and Francios Soumis. Crew pairing at air france. *European Journal of Operational Research*, 97(2):245–259, 1997.

[14] Guy Desaulniers, Jacques Desrosiers, Yvan Dumas, Marius M Solomon, and François Soumis. Daily aircraft routing and scheduling. *Management Science*, 43(6):841–855, 1997.

[15] Shimon Even, Alon Itai, and Adi Shamir. On the complexity of time table and multi-commodity flow problems. In *Foundations of Computer Science, 1975., 16th Annual Symposium on*, pages 184–193. IEEE, 1975.

[16] Thomas A Feo and Jonathan F Bard. Flight scheduling and maintenance base planning. *Management Science*, 35(12):1415–1432, 1989.

[17] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

[18] Balaji Gopalakrishnan and Ellis Johnson. Airline crew scheduling: State-of-the-art. *Annals of Operations Research*, 140:305–337, 2005.

[19] Ram Gopalan and Kalyan T Talluri. The aircraft maintenance routing problem. *Operations Research*, 46(2):260–271, 1998.

[20] Christopher A Hane, Cynthia Barnhart, Ellis L Johnson, Roy E Marsten, George L Nemhauser, and Gabriele Sigismondi. The fleet assignment problem: solving a large-scale integer program. *Mathematical Programming*, 70(1):211–232, 1995.

[21] Karla L Hoffman and Manfred Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6):657–682, 1993.

[22] Timothy L Jacobs, Barry C Smith, and Ellis L Johnson. Incorporating network flow effects into the airline fleet assignment process. *Transportation Science*, 42(4):514–529, 2008.

[23] Diego Klabjan, Ellis L Johnson, George L Nemhauser, Eric Gelman, and Srini Ramaswamy. Airline crew scheduling with time windows and plane-count constraints. *Transportation Science*, 36(3):337–348, 2002.

[24] Diego Klabjan and Karsten Schwan. Airline crew pairing generation in parallel. Technical report, Technical Report TLI/LEC-99-09, Georgia Institute of Technology, Atlanta, GA, 1999.

[25] Shan Lan, John-Paul Clarke, and Cynthia Barnhart. Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions. *Transportation Science*, 40(1):15–28, 2006.

[26] Sylvie Lavoie, Michel Minoux, and Edouard Odier. A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, 35(1):45–58, 1988.

[27] David Levine. Application of a hybrid genetic algorithm to airline crew scheduling. *Computers & Operations Research*, 23(6):547–558, 1996.

[28] Manoj Lohatepanont and Cynthia Barnhart. Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment. *TRANSPORTATION SCIENCE*, 38(1):19–32, 2004.

[29] R Marsten. Crew planning at delta airlines. In *XV Mathematical Programming Symposium. Presentation*, 1994.

[30] Anne Mercier, Jean-François Cordeau, and François Soumis. A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, 32(6):1451–1476, 2005.

[31] Anne Mercier and François Soumis. An integrated aircraft routing, crew scheduling and flight retiming model. *Computers & Operations Research*, 34(8):2251–2265, 2007.

[32] Brian Rexing, Cynthia Barnhart, Tim Kniker, Ahmad Jarrah, and Nirup Krishnamurthy. Airline fleet assignment with time windows. *Transportation Science*, 34(1):1–20, 2000.

[33] Jay M Rosenberger, Ellis L Johnson, and George L Nemhauser. A robust fleet-assignment model with hub isolation and short cycles. *Transportation Science*, 38(3):357–368, 2004.

[34] Sergey Shebalov and Diego Klabjan. Robust airline crew pairing: Move-up crews. *Transportation Science*, 40(3):300–312, 2006.

[35] Kalyan T Talluri. The four-day aircraft maintenance routing problem. *Transportation Science*, 32(1):43–53, 1998.

[36] Pamela H Vance, Cynthia Barnhart, Ellis L Johnson, and George L Nemhauser. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45(2):188–200, 1997.

A Appendix

In this section, detailed and constructive proofs of Theorem 1 and Proposition 12 are given.

Proof of Theorem 1. We first prove that aircraft routing problem can be reduced in linear time to an equivalent equigraph routing problem. Let $T, \mathcal{A}, \mathcal{B}, F, D, \kappa_\alpha^d, \gamma_\alpha^d$ be an instance of the aircraft routing problem.

First, we build the equigraph routing problem instance $G', N'_d, M'_d, D', \kappa_s^{d'}, \gamma_t^{d'}$. Let t be a chronological index: time t_d on day d gives index $\tau \cdot d + t_d$. The total number of airplanes initially in α is $\theta_\alpha^{t_0} = \sum_{o=1}^{D} \kappa_a^o$. As each flight is covered exactly once, the number of airplanes θ_α^t at t in a is obtained by counting the arrivals and the departures.

$$\theta_\alpha^t = \theta_\alpha^{t_0} + |\{\text{arrivals before t}\}| - |\{\text{departures before t}\}| \tag{5}$$

$$= \theta_\alpha^{t_0} + \sum_{\tau=0}^{t} \left[\sum_{f|\alpha_f^a=\alpha, t_f^a=\tau} 1 - \sum_{f|\alpha_f^d=\alpha, t_f^d=\tau} 1 \right] \tag{6}$$

The *aircraft routing directed graph* $G' = (V', A')$ is defined as follows: for each airport $a \in A$ and each time t such that there is at least one departure from α at t, there is a corresponding internal vertex $v'(\alpha, t) \in V'$. For each airport α there is a corresponding *source* $s'_\alpha \in S$ and a *sink* $t'_\alpha \in T$. For each *flight* $f = ((\alpha_f^d, t_f^d, d_f^d), (\alpha_f^a, t_f^a, d_f^a))$, there is a corresponding *flight arc* between $v'(\alpha_f^d, t_f^d, d_f^d)$ and $v'(\alpha_f^a, t_1, d_1)$ where t_1, d_1 is the first time index greater than t_f^a, d_f^a such that there is a flight departure from α in t_1, d_1. If such a t_1 does not exist, then the destination vertex of the flight arc is t'_α. Each vertex v' corresponds to an airport, a time and a day $day(v')$. Let $t_1, t_2, \ldots, t_n$ be the successive departure times at an airport α, for $i \in [n]$, there are $\theta_\alpha^{t_i}$ *ground arcs* between $v'(\alpha, t_i)$ and $v'(\alpha, t_{i+1})$ or t'_a if $i = n$. Night $N'_d = \delta^-(U'_d)$ is the directed cut of events on day greater than d, i.e. $U'_d = \{v' | day(v') > d\}$. Thus, night characterization $U'_{d+1} \subset U'_d$ is satisfied. Maintenance arcs in M_d are night arcs in N_d that are ground arcs in a base b. For all s, t and d, initial and final constraints are equal $\kappa_\alpha^{d\,\prime} = \kappa_s^d$ and $\gamma_t^{d'} = \gamma_t^d$. This process for a small instance of aircraft routing in on Figure 8.

The resulting graph is an *equigraph*. Indeed, sources s' and terminals t' satisfy $d^-(s') = 0$ and $d^+(t') = 0$. Let $v = v'(\alpha, t)$ be an internal vertex, and t_- the former departure time from α, then

$$d^+(v) - d^-(v) = \theta_v^t - \theta_v^{t_-} - \left[\sum_{f|\alpha_f^a=\alpha, t_f^a=t} 1 - \sum_{f|\alpha_f^d=\alpha, t_f^d=t} 1 \right] = 0 \tag{7}$$

which gives the result.

Aircraft routing feasibility implies equigraph routing feasibility. Suppose the aircraft routing admits a solution $\mathcal{S}$. A flight string σ covers a ground arc a if a is between two successive flight arcs in σ. The number of parallel ground arcs is equal to the number or airplanes at the corresponding airport and time, thus each copy can be assigned to one unique flight string σ to form a corresponding path $P'(\sigma)$. $P'(\sigma)$ is feasible because σ is feasible. $\mathcal{P} = \{P(\sigma) | \sigma \in \mathcal{S}\}$ satisfies

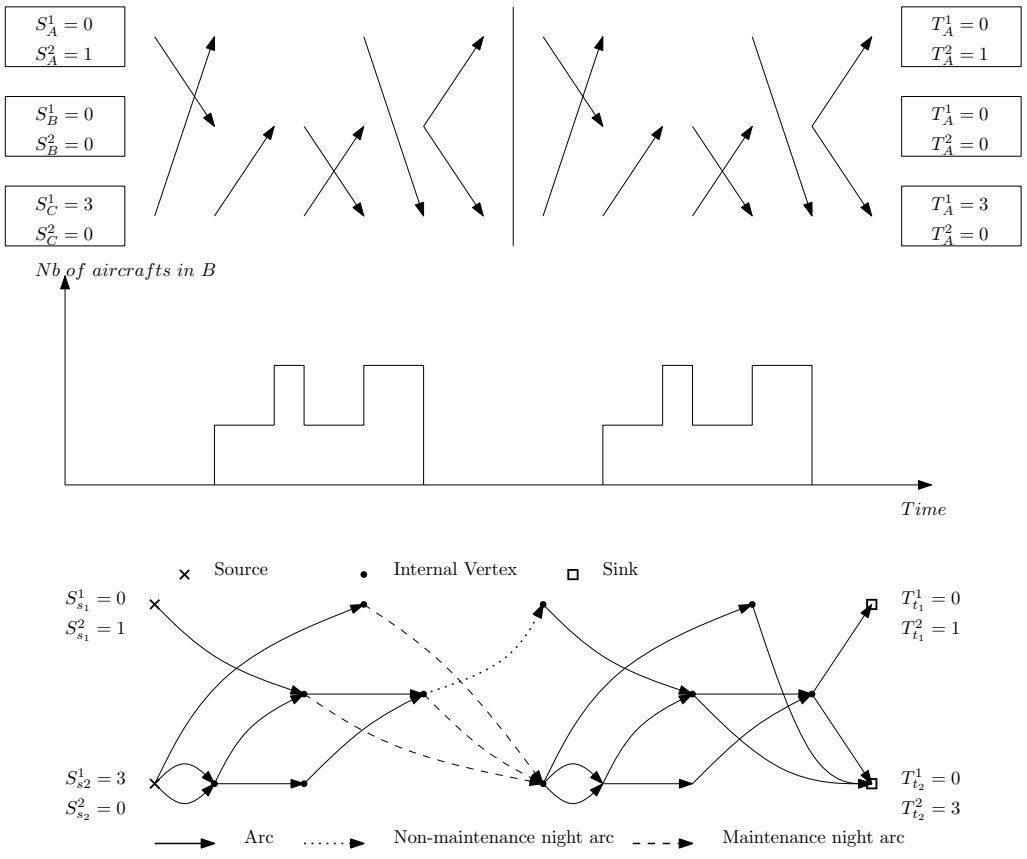

Figure 8: Aircraft routing instance and corresponding equigraph

equigraph cover constraint, initial constraints, and final constraints because R satisfies equigraph cover constraint and terminal constraints.

Conversely, *equigraph routing feasibility implies aircraft routing feasibility.* Let $\mathcal{P}$ be a solution of the equigraph problem. Each feasible path P induces (without ambiguity) a feasible string $S(P)$ and $\sigma = \{S(P), P \in \mathcal{P}\}$ is a feasible routing.

Now, we prove that the equigraph routing problem can be reduced to the aircraft routing problem. Let $G = (V, A), B, D, \kappa_s^d, \gamma_t^d$ be an instance of equigraph routing problem. Then an instance $T', \mathcal{A}', \mathcal{B}', F, D', \kappa_\alpha^{d,}, \gamma_\alpha^{d,}$ of aircraft routing is built as follows. Day $day(v)$ has already been defined for an equigraph with night. A time index respecting the natural ordering of each vertex of a day is obtained thanks to a depth first exploration of the reduced graphs $U_d \backslash U_{d+1}$ corresponding to one day, assigning to each vertex the maximum index of its predecessors plus one. To each arc in $(v_1, v_2) \in A \backslash B$ corresponds a flight $f = ((\alpha(v_1), t(v_1), d(v_1)), (\alpha(v_2), t(v_2), d(v_2))$. T' is the maximum of the time indices.

Aircraft routing feasibility implies equigraph routing feasibility: Feasible paths gives feasible strings. *Equigraph routing feasibility implies implies aircraft routing feasibility:* Feasible strings gives feasible paths. □

Proof of Proposition 12. Let $(y_P) \in [0,1]^P$ be a feasible solution to the linear relaxation of the path formulation (4). To each path P corresponds a unique path $\pi \in \mathcal{G}$. Let $b_{\alpha P}$ be equal to 1 if $\alpha \in \pi(P)$ and 0 otherwise, and $x_\alpha = \sum_{P \in F} b_{\alpha P} y_P$. Then (x_α) satisfies equation $\sum_{\alpha \in \delta^-(\vartheta)} x_\alpha = \sum_{\alpha \in \delta^+(\vartheta)} x_\alpha$ as y_P equally contributes to $\sum_{\alpha \in \delta^-(\vartheta)} x_\alpha$ and $\sum_{\alpha \in \delta^+(\vartheta)} x_\alpha$, and satisfies equation $\sum_{\alpha \in \mathcal{A}_a} x_\alpha = 1$ due to equation $\sum_{\pi \text{ covering } a} y_\pi = 1$. Thus (x_α) is a feasible solution to the linear relaxation of the compact formulation (3).

Let $(x_\alpha)_{\alpha \in \mathcal{A}}$ be a feasible solution to the linear relaxation of formulation (3). Let α_0 be an arc such that x_{α_0} is one of the minima in $A^0 = \{x_\alpha > 0\}$. As equation $\sum_{\alpha \in \delta^-(\vartheta)} x_\alpha = \sum_{\alpha \in \delta^+(\vartheta)} x_\alpha$ is satisfied and x_{α^0} is minimal, if α_0 does not end in a sink (resp. source), there exists α_1^0 among the successors (resp. predecessors) of α^0 such that $x_{\alpha_1^0} > x_{\alpha^0}$. Therefore, by choosing predecessors and successors in a greedy manner, a path π^0 form sources to sinks and satisfying $\alpha^0 \in \pi^0$ and $x_\alpha > x_{\alpha^0}$ for all $\alpha \in \pi^0$ is built. Let $y_{\pi^0} = x_{\alpha^0}$ and

$$x_\alpha^1 = \begin{cases} x_\alpha & if \quad \alpha \notin \pi^0 \\ x_\alpha - x_{\alpha^0} & if \quad \alpha \in \pi^0 \end{cases} \tag{8}$$

As $(x_\alpha^1)_{\alpha \in \mathcal{A}}$ still satisfies equation $\sum_{\alpha \in \delta^-(\vartheta)} x_\alpha = \sum_{\alpha \in \delta^+(\vartheta)} x_\alpha$,we apply the same method to build π^1, then obtain (x_α^2), build π^2 etc. As $x_{\alpha^i} \notin A^j$ for $j > 1$, set $A^k = \emptyset$ after at most $|\mathcal{A}|$ iterations. Besides, (y_{π^i}) satisfies cover constraint $\sum_{\pi \text{ covering } a} y_\pi = 1$ as (x_α) satisfies cover constraint $\sum_{\alpha \in \mathcal{A}_a} x_\alpha = 1$, and (y_π) is a feasible solution of formulation (4) which has the same cost as (x_α), which gives the result. □