

# Aircraft routing: complexity and compact linear integer program

Axel Parmentier, Frédéric Meunier

CERMICS

Aussois – January 6, 2014

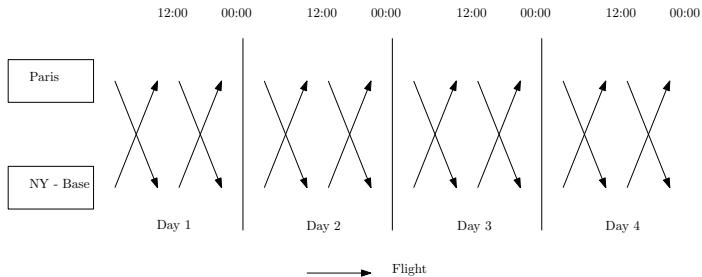
# Table of contents

- 1 Aircraft maintenance routing problem
  - Flight string and aircraft routing
  - Connection graph
- 2 State graph and linear integer program
  - Linear integer program without maintenance constraint
  - Aircraft routing state graph
  - State graph and linear integer programming
- 3 Complexity
  - Routing equigraph
  - Equigraphs properties
  - NP-completeness
  - Fixed parameter tractability

# Aircraft routing problem

Feasible routing

Feasible string

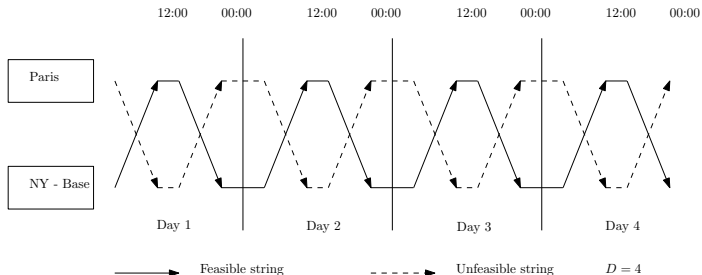


# Aircraft routing problem

## Feasible routing

- Cover constraint

## Feasible string



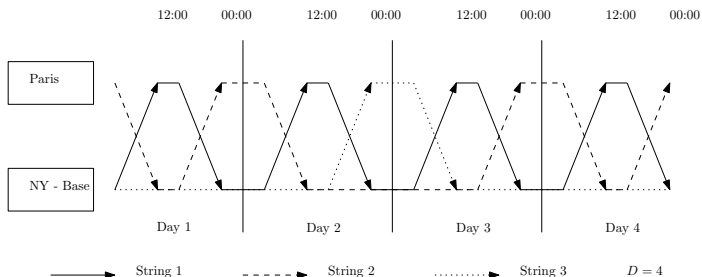
# Aircraft routing problem

## Feasible routing

- Cover constraint
- Initial and final conditions

## Feasible string

- Maintenance constraint



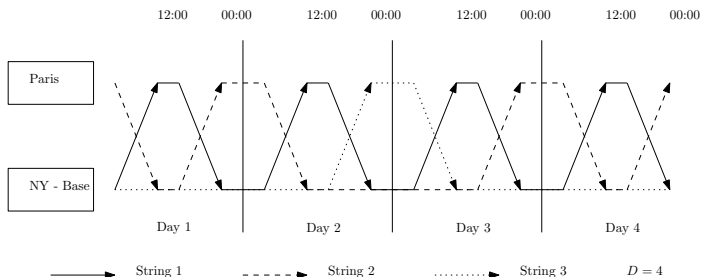
# Aircraft routing problem

## Feasible routing

- Cover constraint
- Initial and final conditions

## Feasible string

- Maintenance constraint



Mathematical formalism: connection graph.

# Connection graph

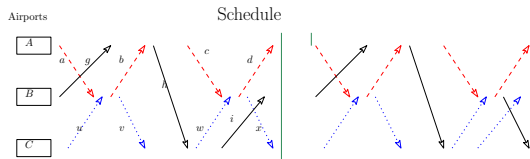
Connection graph

$G = (V, A)$ :

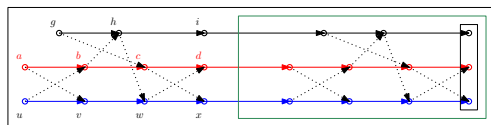
- Flight = vertex  $v$
- Feasible connection = arc  $a$

Maintenance:

- Night = Directed cut  $\delta^-(U)$



Connection graph



○ Flight .....▶ Connection not in partition →▶ Connection in partition

A routing in the connection graph is a partition of the vertices of  $V$  in  $ST$ -paths

# Aircraft routing problem

*Instance:*

- A connection graph  $((S, I, T), A)$
- Night sets  $N_d$  and maintenance night sets  $M_d \subseteq N_d$
- Maintenance constraint  $D$
- Initial and final constraints  $S_s^o, T_t^o$

*Question:*

- Does a feasible routing exist?

## Linear integer program

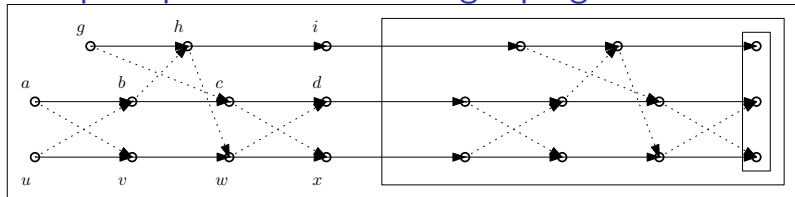
- $|Feas. Connections|D$  variables
- $|Flights|(D + 1)$  constraints

## Complexity results

- NP-complete
- Polynomial when the number of airplanes is given.



# Vertex path partition linear integer program



- Flight ..... ➤ Connection not in partition —➤ Connection in partition

## Lemma

A vertex path partition  $\mathcal{P}$  is entirely determined by connection arcs  $a \in \mathcal{P} \in \mathcal{P}$

- $x_a = 1$  if connection  $a$  is in the partition

$$\sum_{a \in \delta^-(v)} x_a = \sum_{a \in \delta^+(v)} x_a \quad \forall v \in I$$

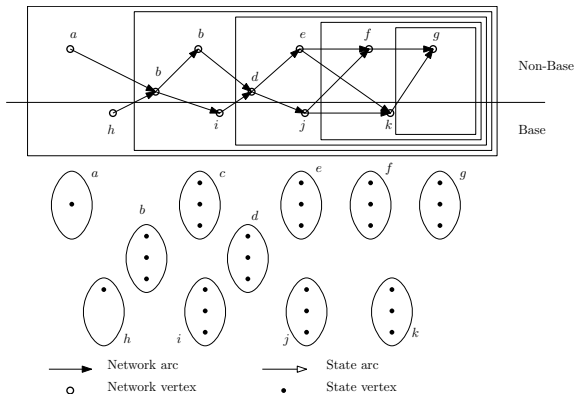
$$\sum_{a \in \delta^-(v)} x_a \geq 1 \quad \forall v \in I \cup T$$

$$\sum_{a \in \delta^+(v)} x_a = 1 \quad \forall v \in S$$

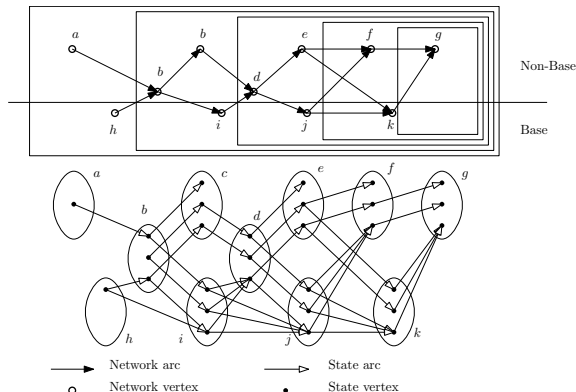
$$x_a \in \{0, 1\} \quad \forall a \in A$$

How can we add constraint in the linear integer program? State graph.

# State graph for aircraft routing



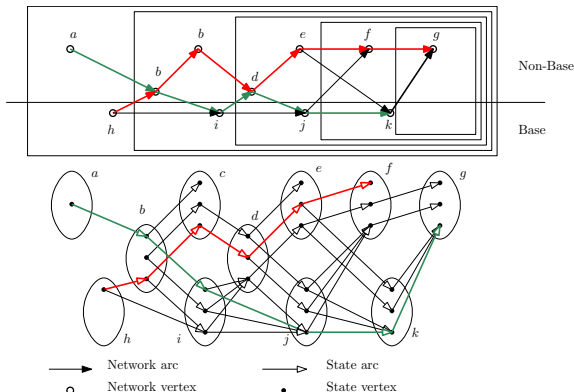
# State graph for aircraft routing



$D$  state vertices for each vertex in the connection graph:

- State vertex = number of day since the last night check.
- State arc = day / (maintenance) night

# State graph for aircraft routing



$D$  state vertices for each vertex in the connection graph:

- State vertex = number of day since the last night check.
- State arc = day / (maintenance) night

## Lemma

A state path induces a feasible path in the connection graph. A partition of the connection graph in state graph paths is a *feasible routing*.

# Aircraft routing linear integer program

- $x_\alpha = 1$  if state arc  $\alpha$  is in the state path partition.

**Theorem (P. & Meunier):**

Same linear relaxation as the “classical” column generation approach.

$$\sum_{\alpha \in \delta^-(\vartheta)} x_\alpha = \sum_{\alpha \in \delta^+(\vartheta)} x_\alpha \quad \forall \vartheta \in \mathcal{V}$$

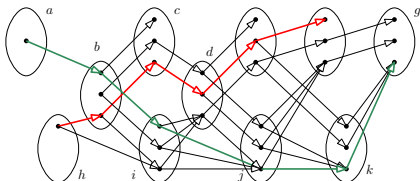
$$\sum_{\substack{\alpha \in \delta^-(\vartheta) \\ \vartheta \in \mathcal{V}_v}} x_\alpha \geq 1 \quad \forall v \in I \cup T$$

$$\sum_{\alpha \in \delta^-(\vartheta)} x_\alpha = 1 \quad \forall \vartheta \in \mathcal{S}$$

$$x_\alpha \in \{0, 1\} \quad \forall \alpha \in \mathcal{A}$$

Optimize passenger connections ( $c_\alpha = c_a$ )

$$\min \sum_{\alpha \in \mathcal{A}} c_\alpha x_\alpha$$



# Complexity

## Theorem (P. & Meunier)

Aircraft routing feasibility problem is NP-complete.

## Theorem (P. & Meunier)

Aircraft routing feasibility problem can be solved in  $O(nD^k)$  where  $n$  is the number of flights and  $k$  is the number of airplanes.

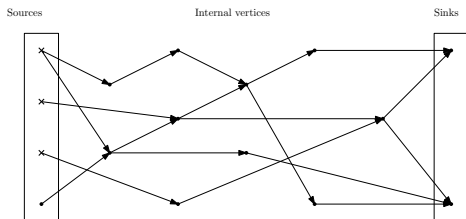
In the connection graph, each airplane is identified: needed for optimization but not for feasibility.

- Introduce the routing equigraph (smaller graph for feasibility only).
- Definition and properties of *equigraph*: guaranty feasibility without identifying aircraft.

## Equigraph definition

An acyclic directed graph is an *equigraph* if its vertices can be partitioned in three sets:

- Sources  $v \in S$  satisfying  $\delta^-(v) = \emptyset$  and  $\delta^+(v) \neq \emptyset$ .
- Internal vertices  $v \in I$  satisfying  $\delta^-(v) = \delta^+(v) > 0$ .
- Sinks  $v \in S$  satisfying  $\delta^-(v) \neq \emptyset$  and  $\delta^+(v) = \emptyset$



# Routing equigraph

Routing equigraph

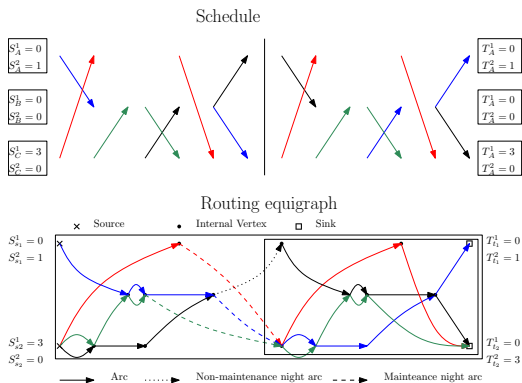
$G = (V, A)$ :

- Flight = arc  $a$
- (airport, time) = vertex  $v$

Maintenance:

- Night = Directed cut  $\delta^-(U)$

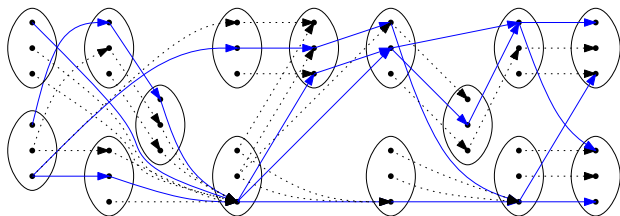
A routing in the routing equigraph is a partition of the arcs of  $A$  in  $ST$ -paths



A digraph is an equigraph if  $\delta^-(v) = \delta^+(v)$  for all internal vertices (a vertex  $v$  is either a source, a sink or an internal vertex).



# State graph on the routing equigraph



$$\sum_{\substack{\alpha \in \delta^-(\vartheta) \\ \vartheta \in \mathcal{V}_v}} x_\alpha = 1, \forall v \in V$$

becomes

$$\sum_{\alpha \in \mathcal{A}_a} x_\alpha = 1, \forall a \in A$$

State path partition gives an equigraph in the state graph.

- How to obtain a path partition of the arcs of an equigraph?
- Searching a feasible equigraph instead of a path partition: faster deterministic algorithm.

# Equigraph partition problem and greedy algorithm

*Instance:*

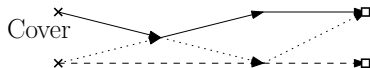
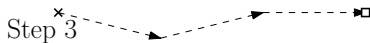
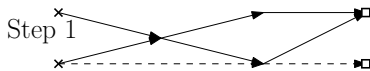
- An equigraph  $((S, I, T), A)$

*Solution:*

- A partition of the arcs of  $G$  in  $S - T$  paths.

## Greedy algorithm lemma

If  $P$  is a path from  $S$  to  $T$ , then  $G \setminus P$  is still an equigraph



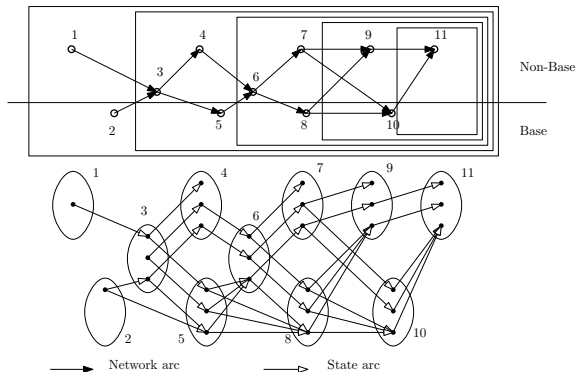
# Fixed parameter tractability

## Theorem (P. & Meunier)

Aircraft routing is fixed parameter tractable in  $D^k$ . It can be solved in  $O(nD^k)$ .

The following algorithm is inspired from Fortune, Hopcroft, and Wyllie, *the directed subgraph homeomorphism problem*, 1980.

- Build a state graph on the routing equigraph
- Order the vertices of the equigraph in a topological ordering



# Pebbling game

## Game goal:

Move  $k$  identical pebbles from the source configuration to the sink configurations set using legal moves.

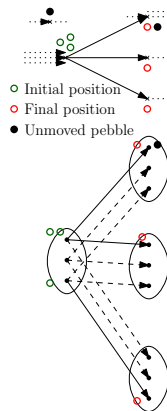
Vertices are enumerated using a topological ordering.

For each vertex  $u$ , one *legal move* is done:

- 1 Pebble initially on  $v_1$  can be moved to  $v_2$  if  $(v_1, v_2)$  is in  $\mathcal{G}$
- 2 Exactly one state arc  $\alpha \in \mathcal{A}_a$  is traversed by a pebble for each  $a \in \delta^-(v)$ .

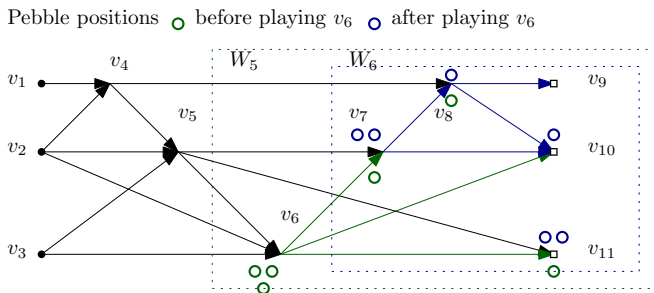
## Lemma

Pebbling game can be won  $\Leftrightarrow$  State graph can be partitioned in paths



## Pebble configurations

Topological ordering: a set of induced subgraphs  $G_i = (V_i, A_i)$  where  $V_i = \{v_j | j \geq i\}$



Number of pebbles on  $v$

- Before playing  $v_i$ :  $|\delta_G^-(v)| - |\delta_{G_i}^-(v)|$
- After playing  $v_i$ :  $|\delta_G^-(v)| - |\delta_{G_{i+1}}^-(v)|$

## Algorithm complexity

The configuration graph  $G_C$  is defined as follows:

- Vertices  $V_C$ : Pebble configurations
- Arcs  $A_C$ : Legal moves.  $|A_C| \leq |V| \cdot D^k$

*A path from the source configuration to the sink configurations set gives a solution to the pebbling game*

### Theorem (P. & Meunier)

Aircraft routing problem is polynomial when  $k$  is fixed. Pebbling game algorithm gives a solution in  $O(nD^k)$

### Proof.

A path finding algorithm is linear in the number of arcs in an acyclic directed graph. □

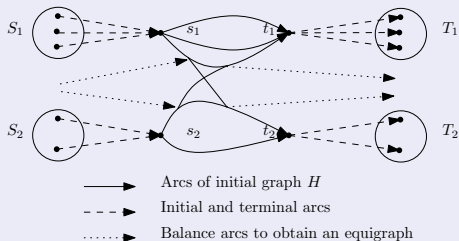
# NP-completeness

## Theorem (P. & Meunier)

Aircraft routing is NP-complete

### Proof.

Reduction from the two commodity arc disjoint paths on acyclic directed graph.



# Questions?

