

# Implementing an Air Taxi System

Daniel Espinoza

Departamento de Ingeniería Industrial, FCFM, Universidad de Chile, Chile

September 5, 2006

- 1 Introduction
- 2 The Routing Problems
- 3 Final Comments

## Work Team

- Mo Bazaraa (Georgia Institute of Technology)
- Emilie Dana (ILOG Inc.)
- Faram Engineer (Georgia Institute of Technology)
- Daniel Espinoza (Universidad de Chile)
- Renan Garcia (Georgia Institute of Technology)
- Marcos Goycoolea (Universidad Adolfo Ibañez)
- Zonghau Gu (ILOG Inc.)
- Alex Khmelnitsky (DayJet)
- George Nemhauser (Georgia Institute of Technology)
- Martin Savelsbergh (Georgia Institute of Technology)
- Eugene Traits (DayJet)

## Per-Seat, On-Demand Jet Services

## History and Current Situation

- Security delays and flight delays.
- Fixed schedules of established air lines.
- “Hub-and-spoke” network configurations.
- New engine technologies, highly fuel efficient.
- New cheap small jet planes for 3-5 passengers.
- Spare capacity on regional airports.

# The Eclipse 500



# The Eclipse 500

- Crew: 2 pilots
- Cruise speed: 694Km/hr
- Range: 2370Km
- Maximum altitude: 12496m
- Passenger Capacity: 3 passengers





## “Per-Seat, On-Demand” Jet Services How to Keep Air Transportation Moving at the Speed of Business



# Business Model

- External factors:
  - On-line demand.
  - Flexible schedules.
  - Last minute coach fares.
  - High service quality.
  - Wide range coverage.
- Internal factors:
  - Dynamic route creation.
  - Fully integrated logistic management.
  - Passenger aggregation.

# Optimization Problems

- On-line accept/reject problem.
- Optimal routing of accepted clients.
- Location of home bases for planes.
- Revenue Management.
- Disruption Recovery.
- Maintenance.
- Demand Estimation.

# Optimization Problems

## Accept/reject Problem

Given a list of accepted requests, and a new request, find a feasible itinerary that cover all requests.

## Optimal Routing

Given a list of accepted requests, find a minimum cost feasible itinerary covering all accepted requests.

# Flight Requests

- Origin.
- Destination.
- Earliest departure time.
- Latest arrival time.
- Number of passengers.
- Total weight.

# Feasible Itineraries

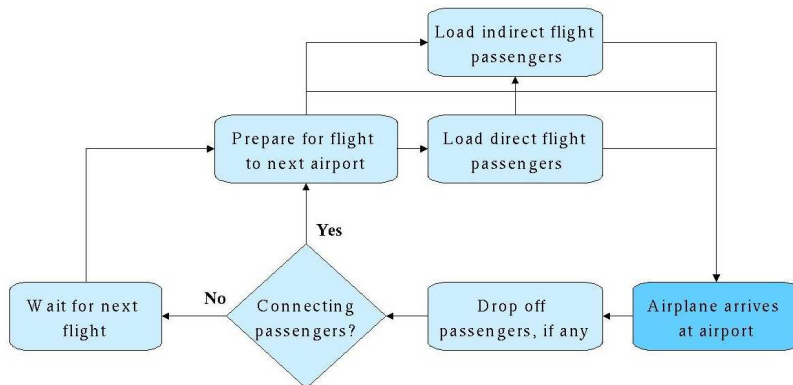
- Two working crews (morning and evening shift).
- Each crew starts and finish in its home base.
- No passenger makes more than one intermediary stop.
- No more than 3 passengers on-board.
- No crew flight for more than 8 hours.
- No crew works for more than 11 hours.
- Total weight of the plane within security limits.

# A Multicommodity flow model with side constraints

- Nodes - Define events that characterize a plane itinerary.
  - Take in a passenger.
  - Drop of a passenger.
  - Flight from airport A to airport B.
  - Wait at some airport A.
- Edges - Connect nodes that might be part of a feasible/optimal itinerary.

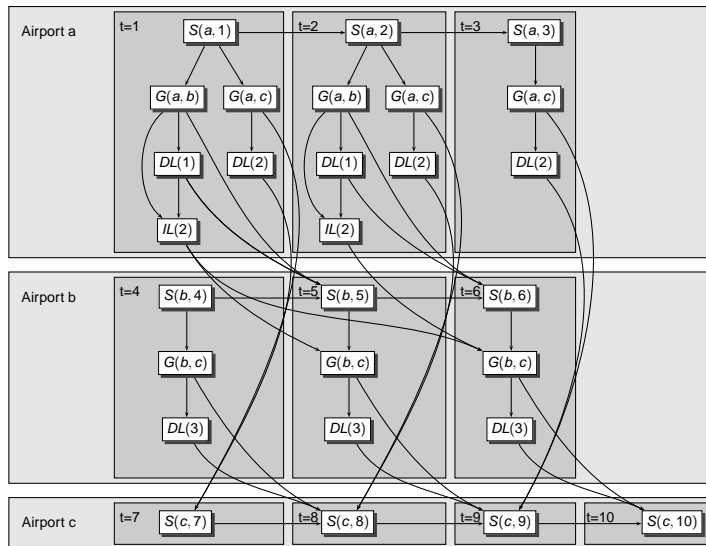
## Modeling the Problem

# A Multicommodity flow model with side constraints



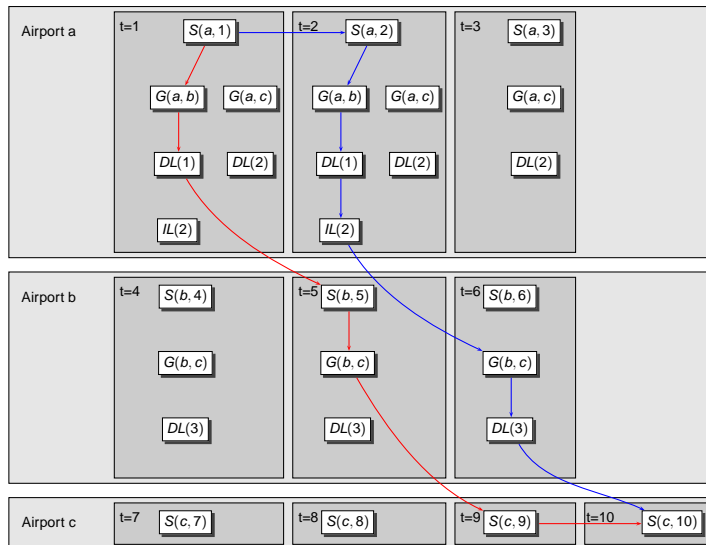
## Modeling the Problem

## An Example



## Modeling the Problem

## Two sample itineraries

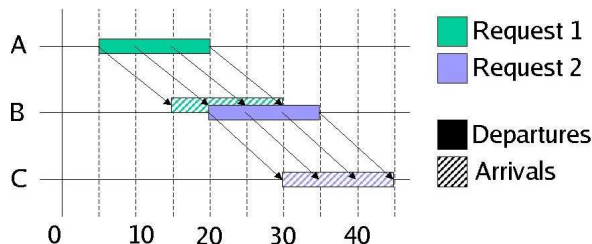


# Obtaining Manageable Problems

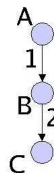
- How do we reduce the network size?
- Can we discard (some) optimal solutions?
- Can we identify edges that do not belong to any optimal solution?
- Can we use side constraints to discard edges?
- Can we approximate the full network?

# The Rolling Forward Algorithm

Many “equivalent” solutions:

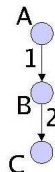
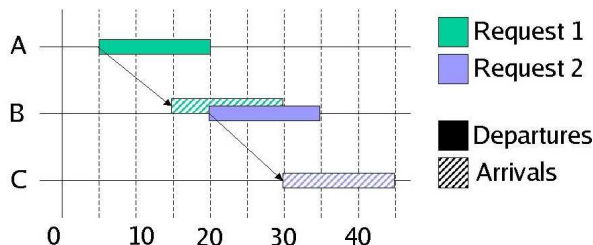


Equivalent Solutions: Same cost, same sequence.  
Different take-off times.



# The Rolling Forward Algorithm

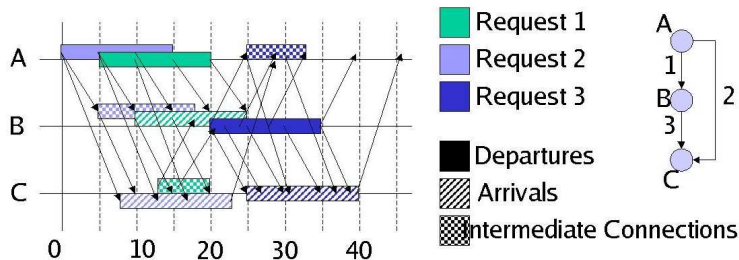
We are content with just Earliest Departure (ED) Solution



(Always takes off as soon as possible)

# The Rolling Forward Algorithm

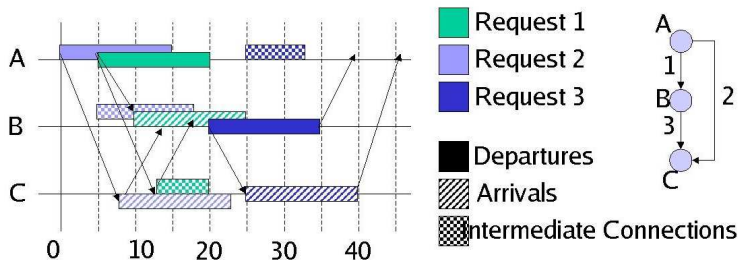
Even small problems can be very complex:



Intermediate connections => many solutions.

# The Rolling Forward Algorithm

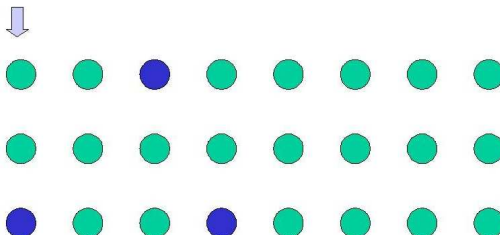
There aren't very many ED solutions



Ideal situation where we can take out all non-necessary edges

# The Rolling Forward Algorithm

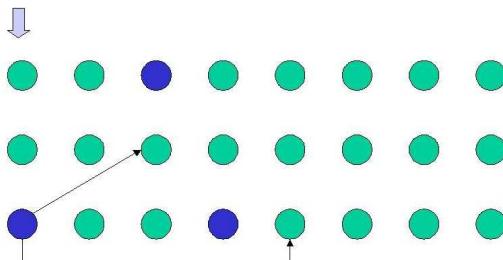
Building a network containing all Earliest Departure (ED) Solutions:



- Mark all Airport/Time nodes that allow to pick-up a request for the first time, and the starting/ending node for the plane in the network.

# The Rolling Forward Algorithm

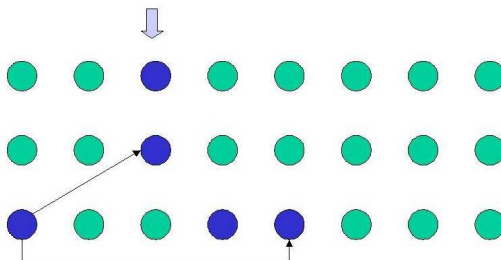
Building a network containing all Earliest Departure (ED) Solutions:



- Process Active Nodes ordered by time and type.
- Add necessary connections for each node.

# The Rolling Forward Algorithm

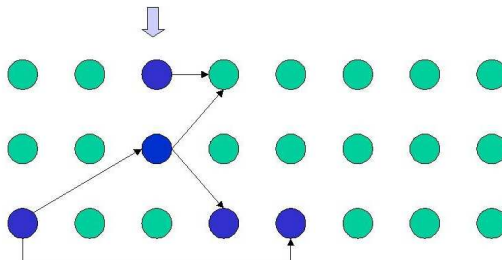
Building a network containing all Earliest Departure (ED) Solutions:



- Process Active Nodes ordered by time and type.
- Add necessary connections for each node.

# The Rolling Forward Algorithm

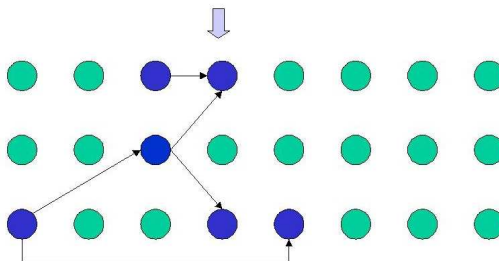
Building a network containing all Earliest Departure (ED) Solutions:



- Process Active Nodes ordered by time and type.
- Add necessary connections for each node.

# The Rolling Forward Algorithm

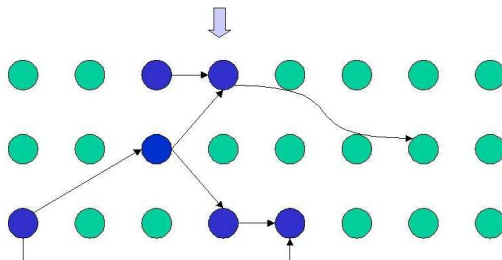
Building a network containing all Earliest Departure (ED) Solutions:



- Process Active Nodes ordered by time and type.
- Add necessary connections for each node.

# The Rolling Forward Algorithm

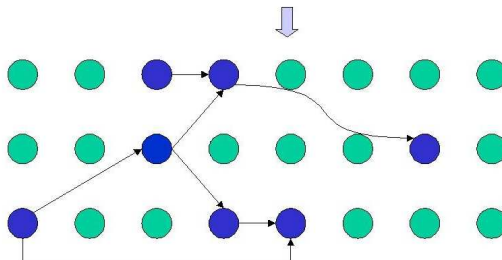
Building a network containing all Earliest Departure (ED) Solutions:



- Process Active Nodes ordered by time and type.
- Add necessary connections for each node.

# The Rolling Forward Algorithm

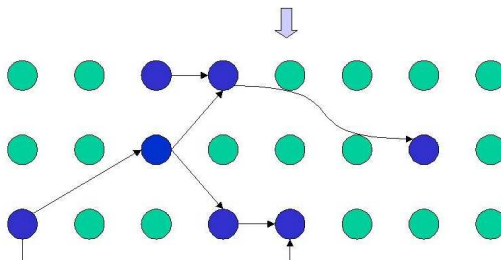
Building a network containing all Earliest Departure (ED) Solutions:



- Process Active Nodes ordered by time and type.
- Add necessary connections for each node.

# The Rolling Forward Algorithm

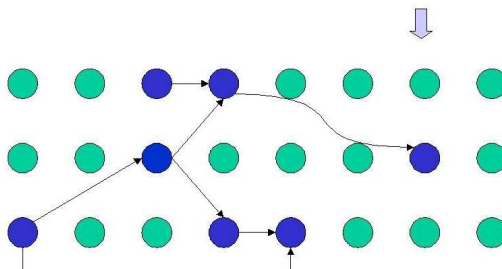
Building a network containing all Earliest Departure (ED) Solutions:



- Process Active Nodes ordered by time and type.
- Add necessary connections for each node.

# The Rolling Forward Algorithm

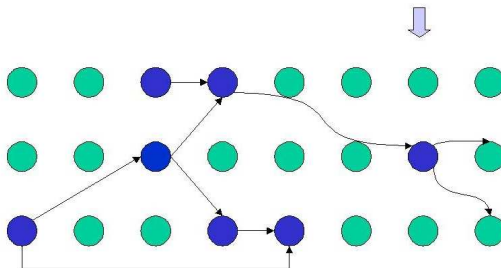
Building a network containing all Earliest Departure (ED) Solutions:



- Process Active Nodes ordered by time and type.
- Add necessary connections for each node.

# The Rolling Forward Algorithm

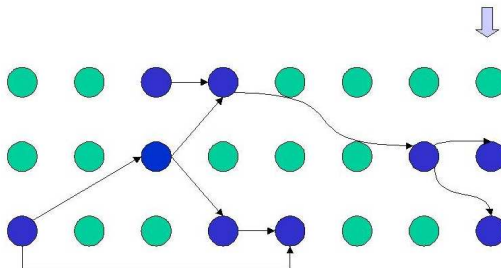
Building a network containing all Earliest Departure (ED) Solutions:



- Process Active Nodes ordered by time and type.
- Add necessary connections for each node.

# The Rolling Forward Algorithm

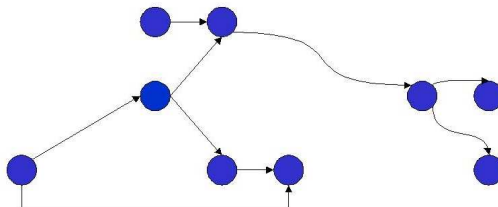
Building a network containing all Earliest Departure (ED) Solutions:



- Process Active Nodes ordered by time and type.
- Add necessary connections for each node.

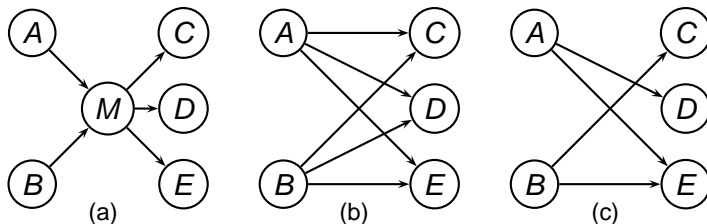
# The Rolling Forward Algorithm

Building a network containing all Earliest Departure (ED) Solutions:



- Process Active Nodes ordered by time and type.
- Add necessary connections for each node.

# The Aggregation Algorithm



- Use a low degree node and replace it by all combination edges.
- Use side constraints to eliminate some.
- Strengthen LP relaxation bound.

# Flexible Time Discretizations

- Use regular discretization.
- Allow for different discretization per airport.
- Allow for non-homogeneous time discretization.
- Can reduce the size of the problem by a factor of 7-8.
- The price of larger discretizations (with special points) is only about 2-3%.

# Reducing the network size

Instance	Initial Graph		Reduced Graph	
	Nodes	Edges	Nodes %	Edges %
R1Y1Q4B	166336	469149	10.93%	66.94%
	171613	485399	10.97%	67.01%
	175245	495579	10.89%	66.92%
R1Y1Q3A	59649	166213	12.16%	68.69%
	58933	160653	11.84%	67.64%
	59437	164120	11.87%	67.34%
R1Y1Q2C	15990	43421	12.01%	67.57%
	15780	43356	12.18%	66.77%
	16408	44927	12.01%	67.52%

# Solving Large Scale Problems

- The model can efficiently solve small problems (3-14 airplanes).
- Expected number of planes: 300-1000 planes.
- Can write down the model for 200 planes.
- Local Search approach.
- Trivially parallelizable.
- Asynchronous parallel model (master-worker approach).

# Computational Results

## 10 CPU, 4 hours (wall time)

flights	deadhead	clients	flight time	Imp.
2667	696	2572	208720	8.4%
2523	637	2597	188289	5.3%
2495	602	2575	187087	5.0%

## 10 CPU, 6 hours (wall time)

flights	deadhead	clients	flight time	Imp.
2637	667	2572	191091	9.6%
2500	614	2597	178277	6.0%
2468	583	2575	177667	5.8%

## Final Comments

- Optimization can help solve large instances.
- Necessary to simplify the problem.
- Identify inactive constraints.
- Integrate scheduling and maintenance.
- Many related problems remain open.