A Recovery Algorithm for a Disrupted Airline Schedule

Niklaus Eggenberg
Matteo Salani and Prof. M. Bierlaire

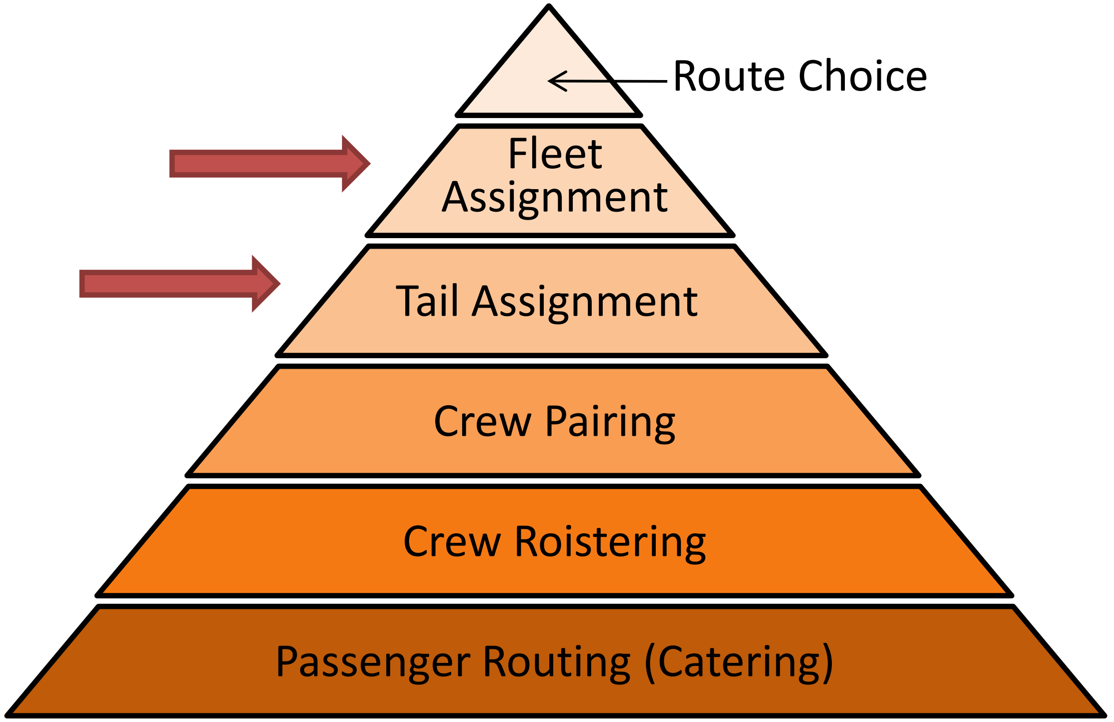
In collaboration with *APM Technologies*

TRANSP-OR
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

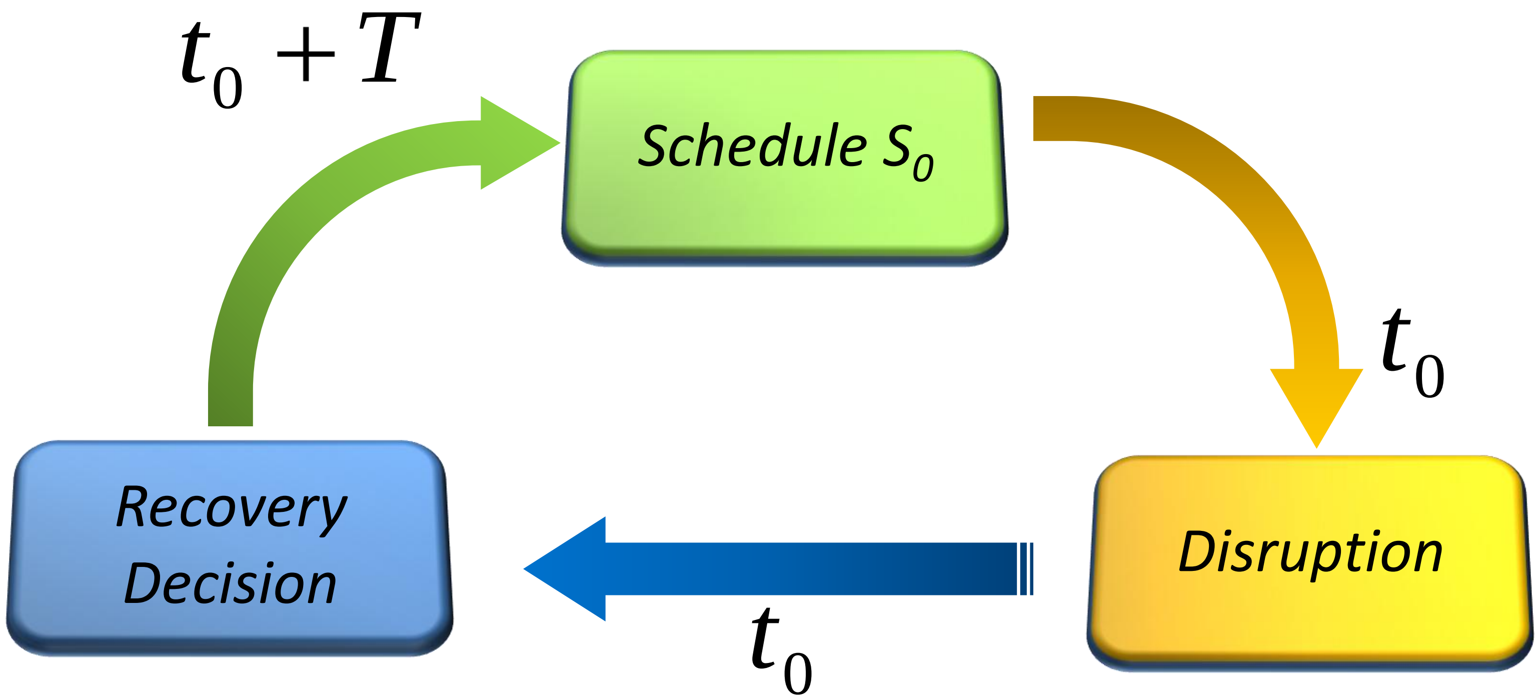
Index

- Airline Scheduling in general
- The Disrupted Schedule Recovery Problem (DSRP)
- The Column Generation (CG) approach
- The Recovery Network Model (RNM)
- The pricing algorithm
- Future Work and Conclusions

Airline Scheduling Approach



Disrupted Schedule Recovery



Definitions

- ***Disruption***

  event making a schedule unrealizable

- ***Recovery***

  action to get back to initial schedule

- ***Recovery Period (T)***

  time needed to recover initial schedule

Definitions

- ***Recovery Plan***

  set of actions to recover disrupted schedule

- ***Recovery Scheme (r)***

  set of actions for a resource (plane)

Hypothesis

- consider only fleet and tail assignment
- no repositioning flights
- no early departure for flights
- work with universal time (UMT)
- initial state of resources are known
- no irregularity until end of recovery period
- maintenance forced by resource consumption

Column Generation

- column = recovery scheme
- recovery scheme r = way to link Initial State to Final State with succession of flights and maintenances
- suppose set of all possible schemes R known
- find optimal combination of schemes

Master Problem (IMP)

$$\min \quad z_{MP} = \sum_{r \in R} c_r x_r + \sum_{f \in F} c_f y_f$$

$$s.c. \quad \sum_{r \in R} \boldsymbol{b}_r^f x_r + y_f = 1 \qquad \forall f \in F$$

$$\sum_{r \in R} \boldsymbol{b}_r^{s^a} x_r = 1 \qquad \forall s^a \in S^a, \forall a \in A$$

$$\sum_{r \in R} \boldsymbol{b}_r^p x_r \leq 1 \qquad \forall p \in P$$

$$x_r \in \{0,1\} \qquad \forall r \in R$$

$$y_f \in \{0,1\} \qquad \forall f \in F$$

Solving the Master Problem

I. Solve IMP with Branch and Bound

II. Solve linear relaxation LP at each node:

- Restrict LP to sub-set $R' \subseteq R$
- Solve RLP
- Find $\boldsymbol{b}_r \in R \setminus R'$ minimizing reduced cost
- Insert column if *r.c. < 0* and resolve RLP

The Pricing Problem

Find column $\boldsymbol{b}_r \in R \setminus R'$ minimizing reduced cost $\tilde{c}_r^p$

$$\min_{r \in R} \tilde{c}_r^p = c_p^r - \sum_{f \in F} \boldsymbol{b}_r^f \lambda_f - \sum_{a \in A} \sum_{s^a \in S^a} \boldsymbol{b}_r^{s^a} \eta_{s^a} + \mathbf{b}_r^p \mu_p \qquad \forall p \in P$$

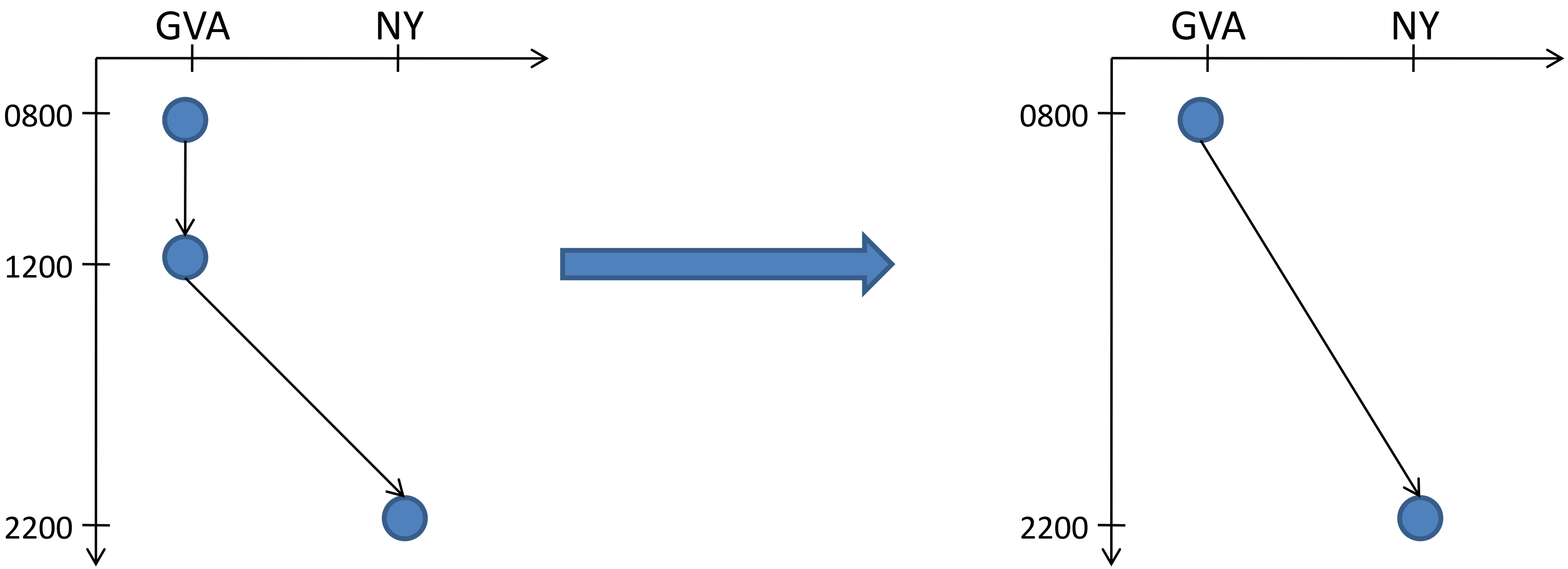
Recovery Network Model

Solve Resource Constrained Elementary Shortest Path Problem (RCESPP)

The Recovery Network (Argüello et al. 97)

- Time-space network $\xrightarrow{x}$ $\downarrow t$
- One network for every plane
- Source node corresponding to initial state
- Sinks corresponding to expected final states
- 3 arc types (NEVER horizontal):
  1. Flight arcs $\longrightarrow$
  2. Maintenance arcs $\Longrightarrow$
  3. Termination arcs (vertical)

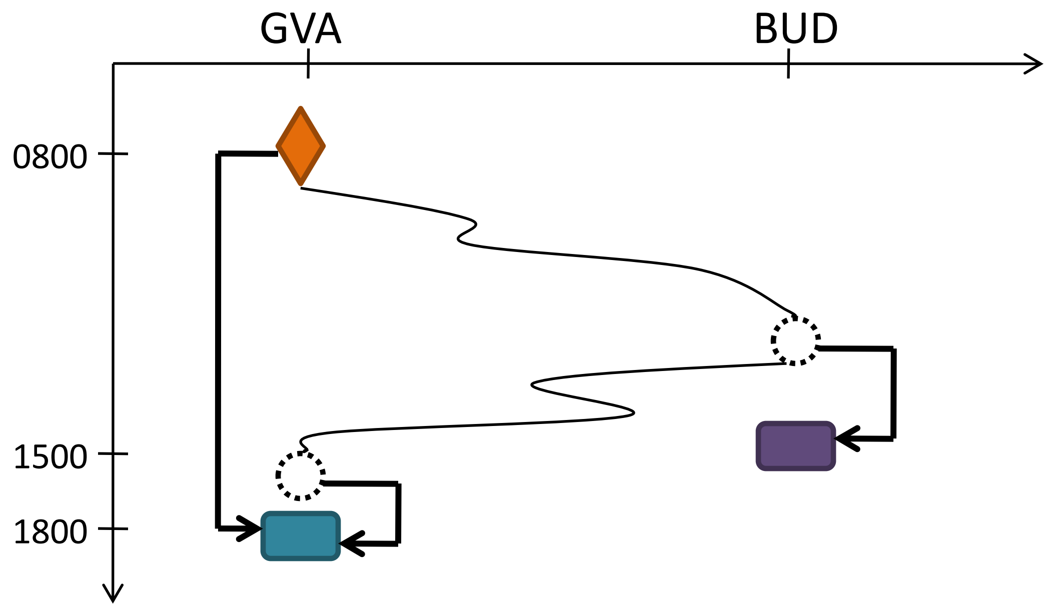
Avoiding Vertical Arcs



Source and Sink Nodes

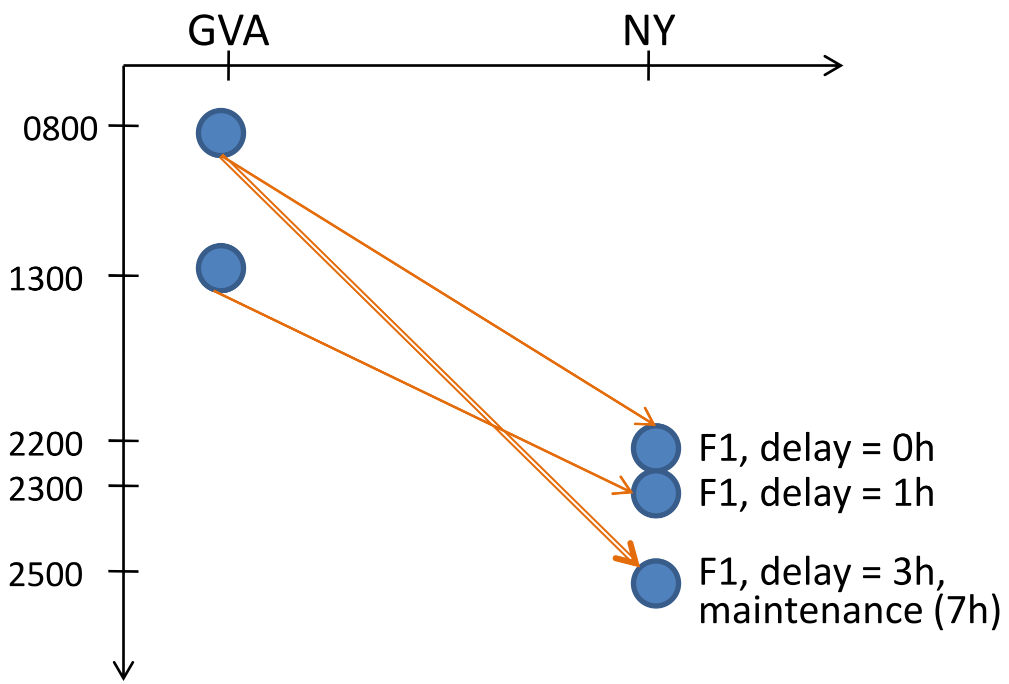
Plane P1, initial state = [GVA, 0800]

Expected States : [GVA, 1800] and [BUD, 1500]



Flight and Maintenance Arcs

flight F1: GVA to NY at 1200



Arc Costs

- Flight arcs: $\longrightarrow$ $c = c^f - \lambda_f$
- Maintenance arcs: $\Longrightarrow$ $c = c^f + c^M - \lambda_f$
- Termination arcs: $c = -\eta_s{}^a$

Recovery Network Properties

- No horizontal arcs
- No vertical arcs except termination arcs
- Node only at earliest availability time
- Grounding time included in arc length (3 types)
- Maintenances are integrated before flight if possible

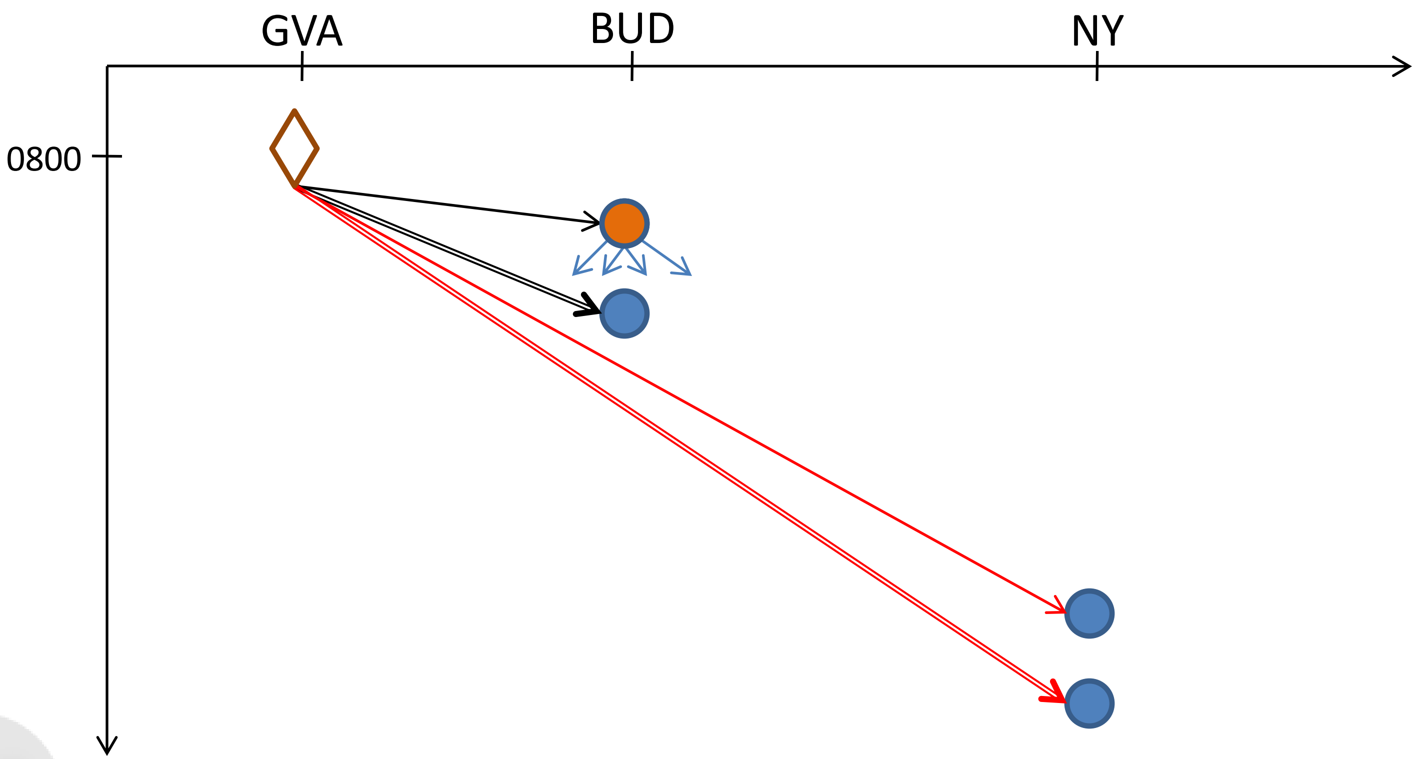
Solving the Pricing Problem

- Find RCESP in Recovery Networks
- 3 phase algorithm:
  1. RN Generation
  2. RN Preprocessing
  3. Solve RCESPP

1. RN Generation

- proceed dynamically: create source, proceed by increasing time
- for all feasible flights create arc and node
- process nodes until all nodes are explored

1. RN Generation



Feasible Flights

- departure airport must match
- departure and landing in airports' activity slots
- control parameters

Control Parameters

- maximum delay bound
- time discretisation (keep earliest availability time)
- minimal time before reaching sinks
- maximal waiting time (D. Messina)

Network Cleaning

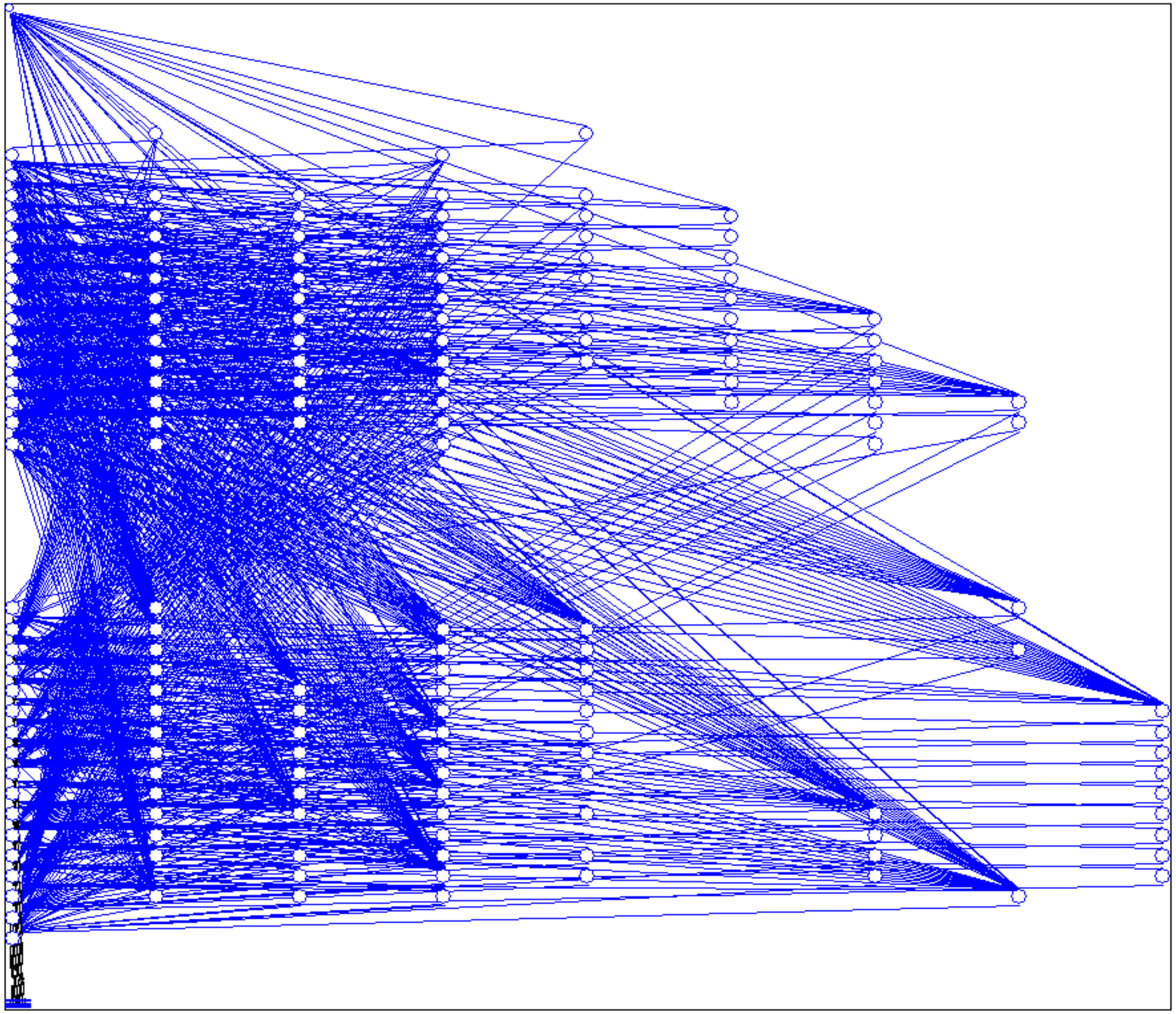
- proceed forwards, remove nodes without predecessors (!source!)
- proceed backwards, remove nodes without successors (!sinks!)

Tested Instance

- 48 flights
- 9 airports
- 5 planes
- T = 2880 minutes

Delay Bound Parameter DB

DB [min]	2880	clean	1440	clean	720	clean	360	clean	180	clean	90	clean
# nodes	2117	1939	1995	1834	1611	1443	965	842	473	412	306	255
# arcs	17825	16791	14851	14301	10836	10366	6773	6507	3348	3247	1986	1916
flight	11775	11258	9780	9518	7069	6843	4278	4158	2056	2009	1205	1181
maint	6050	5533	5071	4783	3767	3523	2495	2349	1292	1238	781	735
# sinks	5	5	5	5	5	5	5	5	5	5	5	5
# term arcs	2780	2780	2780	2780	2610	2610	2120	2120	1335	1335	840	840
time [s]	3.53	1.63	3	0.65	2.24	0.25	1.35	0.11	0.63	0.01	0.39	0.01



Time Discretisation Parameter Δ

Δ [min]	5	clean	15	clean	30	clean	60	clean	120	clean	240	clean
# nodes	2117	1939	781	721	416	386	228	211	141	133	113	106
# arcs	17825	16791	6776	6430	3673	3503	2069	1979	1578	1506	1093	1071
flight	11775	11258	4455	4282	2411	2326	1345	1300	978	950	703	683
maint	6050	5533	2321	2148	1262	1177	724	679	600	556	390	388
# sinks	5	5	5	5	5	5	5	5	5	5	5	5
# term arcs	2780	2780	1055	1055	570	570	325	325	270	270	180	180
time [s]	3.61	1.63	1.26	0.08	0.68	0.02	0.38	0.01	0.3	0.01	0.21	< 0.01

Min Time before Sink Absorption Φ

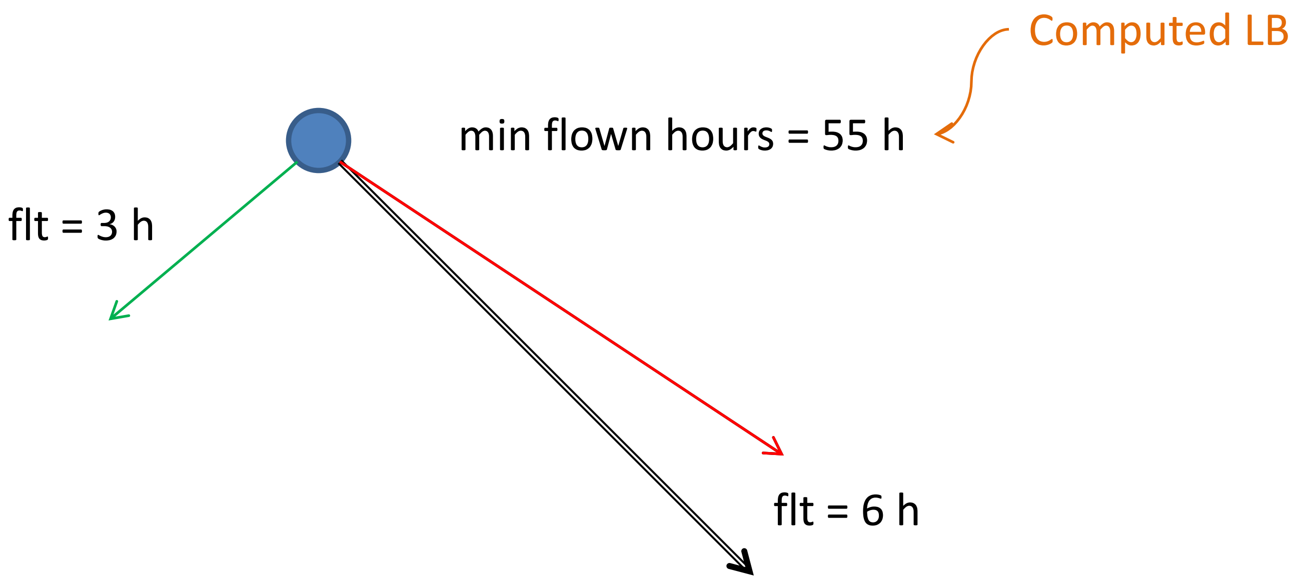
Φ [min]	0	0 clean	1440	clean	2160	clean	2520	clean	2700	clean	2790	clean
# nodes	2117	1939	2117	1939	2117	1939	2117	1939	2117	1758	2117	0
# arcs	17825	16791	17825	16791	17825	16791	17825	16791	17825	14670	17825	0
flight	11775	11258	11775	11258	11775	11258	11775	11258	11775	9797	11775	0
maint	6050	5533	6050	5533	6050	5533	6050	5533	6050	4873	6050	0
# sinks	5	5	5	5	5	5	5	5	5	5	0	0
# term arcs	2780	2780	1630	1630	1050	1050	330	330	25	25	0	0
time [s]	3.6	1.66	3.62	1.63	3.56	1.62	3.59	1.62	3.57	4.03	3.61	0.07

2. RN Preprocessing

- compute upper/lower bounds on costs and resource consumption (spp)
- remove unfeasible arcs according to resources

Example of Arc Removing

max flown hours between maintenances = 60 h



Without Maintenance Slot

% resource used initially	0	clean	25	clean	50	clean	75	clean	100	clean
# nodes	669	386	304	115	70	58	22	1	1	1
# arcs	2586	1949	982	606	165	143	21	0	0	0
flight	2586	1949	982	606	165	143	21	0	0	0
maint	0	0	0	0	0	0	0	0	0	0
# sinks	5	5	4	4	3	3	2	2	1	1
# term arcs	270	270	104	104	39	39	2	2	1	1
time [s]	0.66	0.047	0.22	0.03	0.05	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01

With Maintenance at any Time

% resource used initially	0	clean	25	clean	50	clean	75	clean	100	clean
# nodes	2052	1822	2052	1822	2052	1822	2052	1822	2052	1822
# arcs	16857	15749	16857	15749	16857	15749	16854	15746	16833	15725
flight	11047	10456	11047	10456	11047	10456	11044	10453	11023	10432
maint	5810	5293	5810	5293	5810	5293	5810	5293	5810	5293
# sinks	5	5	5	5	5	5	5	5	5	5
# term arcs	2102	2102	2101	2101	2100	2100	2099	2099	2098	2098
time [s]	3.22	0.234	3.33	0.265	3.36	0.219	3.56	0.25	3.33	0.235

With Maintenance Slot at Night

% resource used initially	0	clean	25	clean	50	clean	75	clean	100	clean
# nodes	737	435	655	338	589	328	572	302	561	302
# arcs	4326	3638	2968	2263	2527	1891	2093	1448	2072	1438
flight	2958	2270	2488	1783	2143	1507	2069	1424	2048	1414
maint	1368	1368	480	480	384	384	24	24	24	24
# sinks	5	5	5	5	5	5	5	5	5	5
# term arcs	572	572	327	327	287	287	196	196	195	195
time [s]	0.84	0.06	0.55	0.08	0.5	0.02	0.39	0.05	0.44	< 0.01

DB = 720, Δ = 15, Φ = 1440

% resource used initially	0	clean	25	clean	50	clean	75	clean	100	clean
# nodes	270	184	222	124	176	118	162	95	151	95
# arcs	1515	1408	957	830	675	602	429	347	408	337
flight	1035	928	732	605	480	407	414	332	393	322
maint	480	480	225	225	195	195	15	15	15	15
# sinks	3	3	3	3	3	3	3	3	3	3
# term arcs	70	70	69	69	69	69	69	69	69	69
time [s]	0.39	0.02	0.19	< 0.01	0.14	0.02	0.01	< 0.01	0.01	< 0.01

3. Solve RCESPP

Use a dynamic programming algorithm proposed by Righini & Salani (2006), which is an extension of Desrochers et al. (1988).

Exploit bounds computed in preprocessing.

Future Work

- Test real instances (next month)
- Explore more widely RCESPP Solver
- Compare solution to real recovery decisions
- Include Algorithm in APM Framework

Conclusions

- Adapted model including maintenances
- Developed 3 phase algorithm to solve PP
- Get quick and easy to parameterize solutions
- Still need real-instance validation

THANKS for your attention

Feel free to ask!