

Airline Planning and Optimisation - AE4423-20

Lecture 1

Introduction, Planning Framework, Demand, Market

Marta Ribeiro
Assistant Professor
Air Transport & Operation
Control & Operations

Delft University of Technology, The Netherlands

November, 2024

Outline - Lecture 1

① Course Information

- ① Course Information
 - a Overview
 - b Content
 - c Staff
 - d Learning Goals
 - e Lectures
 - f Materials
 - g Grading
 - h Workload

② Airline Planning Framework

③ Demand Analysis

- ③ Demand Analysis
 - a Dichotomy
 - b Mismatch
 - c Spoilage & Spillage

④ Demand Models

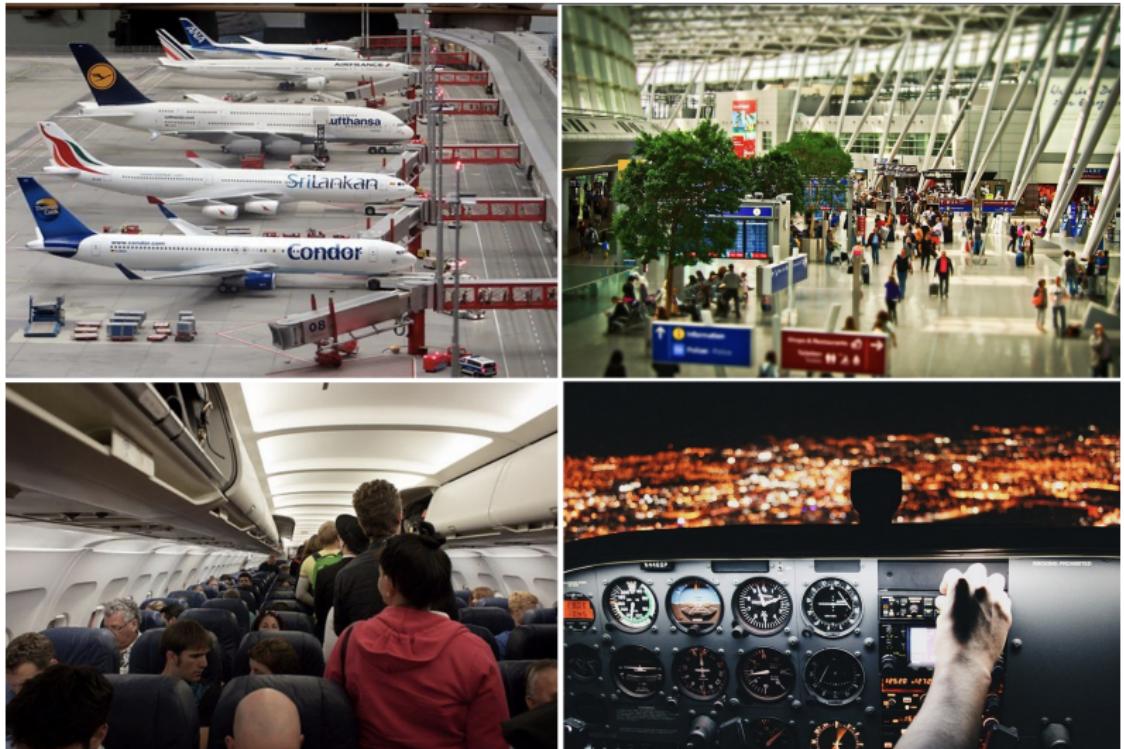
- ④ Demand Models
 - a Time-Series
 - b Causal Methods
 - c Summary

⑤ Market Share

- ⑤ Market Share
 - a S-Curve Model
 - b Quality of Service Index
 - c Logit Model

⑥ Homework

Course Information - Overview



Source: Pixabay

Course Information - Overview (2)

Reality



Lecture

```
14     if (array[order] < array['start_order']) {
15         array = [array['start_order'], array[order], array];
16     }
17
18     start_order = array[0];
19     for (let i = 0; i < array.length - 1; i++) {
20         if (array[i] > array[i + 1]) {
21             let temp = array[i];
22             array[i] = array[i + 1];
23             array[i + 1] = temp;
24         }
25     }
26
27     array.reverse();
28     return array;
29 }
30
31
32 // Sorts arrays by key or value
33 sort_order_by(key) {
34     if (key === 'value') {
35         return array;
36     }
37
38     array.sort((a, b) => a[key] - b[key]);
39
40     return array;
41 }
42
43
44 // Sorts arrays by key or value
45 sort_order_by(key) {
46     if (key === 'value') {
47         return array;
48     }
49
50     array.sort((a, b) => a[key] - b[key]);
51
52     return array;
53 }
54
55
56 // Sorts arrays by key or value
57 sort_order_by(key) {
58     if (key === 'value') {
59         return array;
60     }
61
62     array.sort((a, b) => a[key] - b[key]);
63
64     return array;
65 }
66
67
68 // Sorts arrays by key or value
69 sort_order_by(key) {
70     if (key === 'value') {
71         return array;
72     }
73
74     array.sort((a, b) => a[key] - b[key]);
75
76     return array;
77 }
78
79
80 // Sorts arrays by key or value
81 sort_order_by(key) {
82     if (key === 'value') {
83         return array;
84     }
85
86     array.sort((a, b) => a[key] - b[key]);
87
88     return array;
89 }
90
91
92 // Sorts arrays by key or value
93 sort_order_by(key) {
94     if (key === 'value') {
95         return array;
96     }
97
98     array.sort((a, b) => a[key] - b[key]);
99
100    return array;
101 }
102
103
104 // Sorts arrays by key or value
105 sort_order_by(key) {
106     if (key === 'value') {
107         return array;
108     }
109
110    array.sort((a, b) => a[key] - b[key]);
111
112    return array;
113 }
114
115
116 // Sorts arrays by key or value
117 sort_order_by(key) {
118     if (key === 'value') {
119         return array;
120     }
121
122    array.sort((a, b) => a[key] - b[key]);
123
124    return array;
125 }
126
127
128 // Sorts arrays by key or value
129 sort_order_by(key) {
130     if (key === 'value') {
131         return array;
132     }
133
134    array.sort((a, b) => a[key] - b[key]);
135
136    return array;
137 }
138
139
140 // Sorts arrays by key or value
141 sort_order_by(key) {
142     if (key === 'value') {
143         return array;
144     }
145
146    array.sort((a, b) => a[key] - b[key]);
147
148    return array;
149 }
150
151
152 // Sorts arrays by key or value
153 sort_order_by(key) {
154     if (key === 'value') {
155         return array;
156     }
157
158    array.sort((a, b) => a[key] - b[key]);
159
160    return array;
161 }
162
163
164 // Sorts arrays by key or value
165 sort_order_by(key) {
166     if (key === 'value') {
167         return array;
168     }
169
170    array.sort((a, b) => a[key] - b[key]);
171
172    return array;
173 }
174
175
176 // Sorts arrays by key or value
177 sort_order_by(key) {
178     if (key === 'value') {
179         return array;
180     }
181
182    array.sort((a, b) => a[key] - b[key]);
183
184    return array;
185 }
186
187
188 // Sorts arrays by key or value
189 sort_order_by(key) {
190     if (key === 'value') {
191         return array;
192     }
193
194    array.sort((a, b) => a[key] - b[key]);
195
196    return array;
197 }
198
199
200 // Sorts arrays by key or value
201 sort_order_by(key) {
202     if (key === 'value') {
203         return array;
204     }
205
206    array.sort((a, b) => a[key] - b[key]);
207
208    return array;
209 }
210
211
212 // Sorts arrays by key or value
213 sort_order_by(key) {
214     if (key === 'value') {
215         return array;
216     }
217
218    array.sort((a, b) => a[key] - b[key]);
219
220    return array;
221 }
222
223
224 // Sorts arrays by key or value
225 sort_order_by(key) {
226     if (key === 'value') {
227         return array;
228     }
229
230    array.sort((a, b) => a[key] - b[key]);
231
232    return array;
233 }
234
235
236 // Sorts arrays by key or value
237 sort_order_by(key) {
238     if (key === 'value') {
239         return array;
240     }
241
242    array.sort((a, b) => a[key] - b[key]);
243
244    return array;
245 }
246
247
248 // Sorts arrays by key or value
249 sort_order_by(key) {
250     if (key === 'value') {
251         return array;
252     }
253
254    array.sort((a, b) => a[key] - b[key]);
255
256    return array;
257 }
258
259
260 // Sorts arrays by key or value
261 sort_order_by(key) {
262     if (key === 'value') {
263         return array;
264     }
265
266    array.sort((a, b) => a[key] - b[key]);
267
268    return array;
269 }
270
271
272 // Sorts arrays by key or value
273 sort_order_by(key) {
274     if (key === 'value') {
275         return array;
276     }
277
278    array.sort((a, b) => a[key] - b[key]);
279
280    return array;
281 }
282
283
284 // Sorts arrays by key or value
285 sort_order_by(key) {
286     if (key === 'value') {
287         return array;
288     }
289
290    array.sort((a, b) => a[key] - b[key]);
291
292    return array;
293 }
294
295
296 // Sorts arrays by key or value
297 sort_order_by(key) {
298     if (key === 'value') {
299         return array;
300     }
301
302    array.sort((a, b) => a[key] - b[key]);
303
304    return array;
305 }
306
307
308 // Sorts arrays by key or value
309 sort_order_by(key) {
310     if (key === 'value') {
311         return array;
312     }
313
314    array.sort((a, b) => a[key] - b[key]);
315
316    return array;
317 }
318
319
320 // Sorts arrays by key or value
321 sort_order_by(key) {
322     if (key === 'value') {
323         return array;
324     }
325
326    array.sort((a, b) => a[key] - b[key]);
327
328    return array;
329 }
330
331
332 // Sorts arrays by key or value
333 sort_order_by(key) {
334     if (key === 'value') {
335         return array;
336     }
337
338    array.sort((a, b) => a[key] - b[key]);
339
340    return array;
341 }
342
343
344 // Sorts arrays by key or value
345 sort_order_by(key) {
346     if (key === 'value') {
347         return array;
348     }
349
350    array.sort((a, b) => a[key] - b[key]);
351
352    return array;
353 }
354
355
356 // Sorts arrays by key or value
357 sort_order_by(key) {
358     if (key === 'value') {
359         return array;
360     }
361
362    array.sort((a, b) => a[key] - b[key]);
363
364    return array;
365 }
366
367
368 // Sorts arrays by key or value
369 sort_order_by(key) {
370     if (key === 'value') {
371         return array;
372     }
373
374    array.sort((a, b) => a[key] - b[key]);
375
376    return array;
377 }
378
379
380 // Sorts arrays by key or value
381 sort_order_by(key) {
382     if (key === 'value') {
383         return array;
384     }
385
386    array.sort((a, b) => a[key] - b[key]);
387
388    return array;
389 }
390
391
392 // Sorts arrays by key or value
393 sort_order_by(key) {
394     if (key === 'value') {
395         return array;
396     }
397
398    array.sort((a, b) => a[key] - b[key]);
399
400    return array;
401 }
402
403
404 // Sorts arrays by key or value
405 sort_order_by(key) {
406     if (key === 'value') {
407         return array;
408     }
409
410    array.sort((a, b) => a[key] - b[key]);
411
412    return array;
413 }
414
415
416 // Sorts arrays by key or value
417 sort_order_by(key) {
418     if (key === 'value') {
419         return array;
420     }
421
422    array.sort((a, b) => a[key] - b[key]);
423
424    return array;
425 }
426
427
428 // Sorts arrays by key or value
429 sort_order_by(key) {
430     if (key === 'value') {
431         return array;
432     }
433
434    array.sort((a, b) => a[key] - b[key]);
435
436    return array;
437 }
438
439
440 // Sorts arrays by key or value
441 sort_order_by(key) {
442     if (key === 'value') {
443         return array;
444     }
445
446    array.sort((a, b) => a[key] - b[key]);
447
448    return array;
449 }
450
451
452 // Sorts arrays by key or value
453 sort_order_by(key) {
454     if (key === 'value') {
455         return array;
456     }
457
458    array.sort((a, b) => a[key] - b[key]);
459
460    return array;
461 }
462
463
464 // Sorts arrays by key or value
465 sort_order_by(key) {
466     if (key === 'value') {
467         return array;
468     }
469
470    array.sort((a, b) => a[key] - b[key]);
471
472    return array;
473 }
474
475
476 // Sorts arrays by key or value
477 sort_order_by(key) {
478     if (key === 'value') {
479         return array;
480     }
481
482    array.sort((a, b) => a[key] - b[key]);
483
484    return array;
485 }
486
487
488 // Sorts arrays by key or value
489 sort_order_by(key) {
490     if (key === 'value') {
491         return array;
492     }
493
494    array.sort((a, b) => a[key] - b[key]);
495
496    return array;
497 }
498
499
500 // Sorts arrays by key or value
501 sort_order_by(key) {
502     if (key === 'value') {
503         return array;
504     }
505
506    array.sort((a, b) => a[key] - b[key]);
507
508    return array;
509 }
510
511
512 // Sorts arrays by key or value
513 sort_order_by(key) {
514     if (key === 'value') {
515         return array;
516     }
517
518    array.sort((a, b) => a[key] - b[key]);
519
520    return array;
521 }
522
523
524 // Sorts arrays by key or value
525 sort_order_by(key) {
526     if (key === 'value') {
527         return array;
528     }
529
530    array.sort((a, b) => a[key] - b[key]);
531
532    return array;
533 }
534
535
536 // Sorts arrays by key or value
537 sort_order_by(key) {
538     if (key === 'value') {
539         return array;
540     }
541
542    array.sort((a, b) => a[key] - b[key]);
543
544    return array;
545 }
546
547
548 // Sorts arrays by key or value
549 sort_order_by(key) {
550     if (key === 'value') {
551         return array;
552     }
553
554    array.sort((a, b) => a[key] - b[key]);
555
556    return array;
557 }
558
559
560 // Sorts arrays by key or value
561 sort_order_by(key) {
562     if (key === 'value') {
563         return array;
564     }
565
566    array.sort((a, b) => a[key] - b[key]);
567
568    return array;
569 }
570
571
572 // Sorts arrays by key or value
573 sort_order_by(key) {
574     if (key === 'value') {
575         return array;
576     }
577
578    array.sort((a, b) => a[key] - b[key]);
579
580    return array;
581 }
582
583
584 // Sorts arrays by key or value
585 sort_order_by(key) {
586     if (key === 'value') {
587         return array;
588     }
589
590    array.sort((a, b) => a[key] - b[key]);
591
592    return array;
593 }
594
595
596 // Sorts arrays by key or value
597 sort_order_by(key) {
598     if (key === 'value') {
599         return array;
600     }
601
602    array.sort((a, b) => a[key] - b[key]);
603
604    return array;
605 }
606
607
608 // Sorts arrays by key or value
609 sort_order_by(key) {
610     if (key === 'value') {
611         return array;
612     }
613
614    array.sort((a, b) => a[key] - b[key]);
615
616    return array;
617 }
618
619
620 // Sorts arrays by key or value
621 sort_order_by(key) {
622     if (key === 'value') {
623         return array;
624     }
625
626    array.sort((a, b) => a[key] - b[key]);
627
628    return array;
629 }
630
631
632 // Sorts arrays by key or value
633 sort_order_by(key) {
634     if (key === 'value') {
635         return array;
636     }
637
638    array.sort((a, b) => a[key] - b[key]);
639
640    return array;
641 }
642
643
644 // Sorts arrays by key or value
645 sort_order_by(key) {
646     if (key === 'value') {
647         return array;
648     }
649
650    array.sort((a, b) => a[key] - b[key]);
651
652    return array;
653 }
654
655
656 // Sorts arrays by key or value
657 sort_order_by(key) {
658     if (key === 'value') {
659         return array;
660     }
661
662    array.sort((a, b) => a[key] - b[key]);
663
664    return array;
665 }
666
667
668 // Sorts arrays by key or value
669 sort_order_by(key) {
670     if (key === 'value') {
671         return array;
672     }
673
674    array.sort((a, b) => a[key] - b[key]);
675
676    return array;
677 }
678
679
680 // Sorts arrays by key or value
681 sort_order_by(key) {
682     if (key === 'value') {
683         return array;
684     }
685
686    array.sort((a, b) => a[key] - b[key]);
687
688    return array;
689 }
690
691
692 // Sorts arrays by key or value
693 sort_order_by(key) {
694     if (key === 'value') {
695         return array;
696     }
697
698    array.sort((a, b) => a[key] - b[key]);
699
700    return array;
701 }
702
703
704 // Sorts arrays by key or value
705 sort_order_by(key) {
706     if (key === 'value') {
707         return array;
708     }
709
710    array.sort((a, b) => a[key] - b[key]);
711
712    return array;
713 }
714
715
716 // Sorts arrays by key or value
717 sort_order_by(key) {
718     if (key === 'value') {
719         return array;
720     }
721
722    array.sort((a, b) => a[key] - b[key]);
723
724    return array;
725 }
726
727
728 // Sorts arrays by key or value
729 sort_order_by(key) {
730     if (key === 'value') {
731         return array;
732     }
733
734    array.sort((a, b) => a[key] - b[key]);
735
736    return array;
737 }
738
739
740 // Sorts arrays by key or value
741 sort_order_by(key) {
742     if (key === 'value') {
743         return array;
744     }
745
746    array.sort((a, b) => a[key] - b[key]);
747
748    return array;
749 }
750
751
752 // Sorts arrays by key or value
753 sort_order_by(key) {
754     if (key === 'value') {
755         return array;
756     }
757
758    array.sort((a, b) => a[key] - b[key]);
759
760    return array;
761 }
762
763
764 // Sorts arrays by key or value
765 sort_order_by(key) {
766     if (key === 'value') {
767         return array;
768     }
769
770    array.sort((a, b) => a[key] - b[key]);
771
772    return array;
773 }
774
775
776 // Sorts arrays by key or value
777 sort_order_by(key) {
778     if (key === 'value') {
779         return array;
780     }
781
782    array.sort((a, b) => a[key] - b[key]);
783
784    return array;
785 }
786
787
788 // Sorts arrays by key or value
789 sort_order_by(key) {
790     if (key === 'value') {
791         return array;
792     }
793
794    array.sort((a, b) => a[key] - b[key]);
795
796    return array;
797 }
798
799
799 // Sorts arrays by key or value
800 sort_order_by(key) {
801     if (key === 'value') {
802         return array;
803     }
804
805    array.sort((a, b) => a[key] - b[key]);
806
807    return array;
808 }
809
810
811 // Sorts arrays by key or value
812 sort_order_by(key) {
813     if (key === 'value') {
814         return array;
815     }
816
817    array.sort((a, b) => a[key] - b[key]);
818
819    return array;
820 }
821
822
823 // Sorts arrays by key or value
824 sort_order_by(key) {
825     if (key === 'value') {
826         return array;
827     }
828
829    array.sort((a, b) => a[key] - b[key]);
830
831    return array;
832 }
833
834
835 // Sorts arrays by key or value
836 sort_order_by(key) {
837     if (key === 'value') {
838         return array;
839     }
840
841    array.sort((a, b) => a[key] - b[key]);
842
843    return array;
844 }
845
846
847 // Sorts arrays by key or value
848 sort_order_by(key) {
849     if (key === 'value') {
850         return array;
851     }
852
853    array.sort((a, b) => a[key] - b[key]);
854
855    return array;
856 }
857
858
859 // Sorts arrays by key or value
860 sort_order_by(key) {
861     if (key === 'value') {
862         return array;
863     }
864
865    array.sort((a, b) => a[key] - b[key]);
866
867    return array;
868 }
869
870
871 // Sorts arrays by key or value
872 sort_order_by(key) {
873     if (key === 'value') {
874         return array;
875     }
876
877    array.sort((a, b) => a[key] - b[key]);
878
879    return array;
880 }
881
882
883 // Sorts arrays by key or value
884 sort_order_by(key) {
885     if (key === 'value') {
886         return array;
887     }
888
889    array.sort((a, b) => a[key] - b[key]);
890
891    return array;
892 }
893
894
895 // Sorts arrays by key or value
896 sort_order_by(key) {
897     if (key === 'value') {
898         return array;
899     }
900
901    array.sort((a, b) => a[key] - b[key]);
902
903    return array;
904 }
905
906
907 // Sorts arrays by key or value
908 sort_order_by(key) {
909     if (key === 'value') {
910         return array;
911     }
912
913    array.sort((a, b) => a[key] - b[key]);
914
915    return array;
916 }
917
918
919 // Sorts arrays by key or value
920 sort_order_by(key) {
921     if (key === 'value') {
922         return array;
923     }
924
925    array.sort((a, b) => a[key] - b[key]);
926
927    return array;
928 }
929
930
931 // Sorts arrays by key or value
932 sort_order_by(key) {
933     if (key === 'value') {
934         return array;
935     }
936
937    array.sort((a, b) => a[key] - b[key]);
938
939    return array;
940 }
941
942
943 // Sorts arrays by key or value
944 sort_order_by(key) {
945     if (key === 'value') {
946         return array;
947     }
948
949    array.sort((a, b) => a[key] - b[key]);
950
951    return array;
952 }
953
954
955 // Sorts arrays by key or value
956 sort_order_by(key) {
957     if (key === 'value') {
958         return array;
959     }
960
961    array.sort((a, b) => a[key] - b[key]);
962
963    return array;
964 }
965
966
967 // Sorts arrays by key or value
968 sort_order_by(key) {
969     if (key === 'value') {
970         return array;
971     }
972
973    array.sort((a, b) => a[key] - b[key]);
974
975    return array;
976 }
977
978
979 // Sorts arrays by key or value
980 sort_order_by(key) {
981     if (key === 'value') {
982         return array;
983     }
984
985    array.sort((a, b) => a[key] - b[key]);
986
987    return array;
988 }
989
990
991 // Sorts arrays by key or value
992 sort_order_by(key) {
993     if (key === 'value') {
994         return array;
995     }
996
997    array.sort((a, b) => a[key] - b[key]);
998
999    return array;
1000 }
```

Modelling

Course Information - Content

Management

Planning Framework

Performance Indicators

Demand and Supply

Market share

Tactical Planning

Passenger Mix Flow

Fleet Assignment

Aircraft Rotation

Crew Scheduling

Maintenance



Strategical Planing

Network Structure

Demand Forecast

Network Planning

Fleet Planning

Operations Research

MILP Models

Multi-commodity Flow Problems

Shortest-path Algorithms

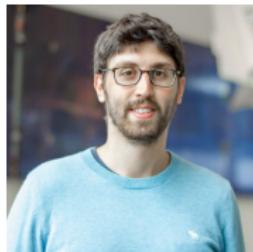
Column Generation Algorithm

Dynamic Programming

Course Information - Lecturers



Marta Ribeiro
Assistant Professor
M.J.Ribeiro@tudelft.nl



Alessandro Bombelli
Assistant Professor
A.Bombelli@tudelft.nl

Lecture **Column Generation Algorithm**



Pieter-Jan Proesmans
Assistant Professor
P.Proesmans@tudelft.nl

Lecture **Future Developments**

Course Information - Learning Goals

At the end of the course, students should be able to:

- ① Define the main strategic, tactical, and operational problems of an airline.
- ② Discuss multiple modeling and solution techniques.
- ③ Formulate an airline strategic and tactical planning problem in a way that it can be solved.
- ④ Evaluate the implications of airline planning decisions.

Course Information - Lectures

Week	Day	Topic
2.1	11 th Nov	Introduction, Planning Framework, Demand, Market
	14 th Nov	Multi-commodity Flow Problems
2.2	18 th Nov	Network and Fleet Planning
	21 st Nov	Passenger Mix Flow
2.3	25 th Nov	Schedule Design & Fleet Assignment
	28 th Nov	Aircraft Routing & Dynamic Programming
2.4	2 nd Dec	Assignment 1
	5 th Dec	Column Generation Algorithm
2.5	9 th Dec	Crew Scheduling
	12 th Dec	Assignment 1
2.6	16 th Dec	Maintenance Scheduling
	19 th Dec	Practice Exam
2.7	6 th Jan	Assignment 2
	9 th Jan	Future Developments

Course Information - Materials

- Software: Python with Gurobi ([Install](#))
- Book: 'The Global Airline Industry', John Wiley & Sons Ltd., Eds. P. Belobaba, A. Odoni, C. Barnhart (2011)
- **Outside of the classroom:**
 - Online videos available
 - Contact per Email

Course Information - Grading

- Final grade: Assignment (60%) + Exam (40%)
- Assignment and Exam must be passed with **5.5 or above**.
- Assignment divide into two parts. Work in **groups of 3 students**, register in Brightspace → Collaboration → Groups.

Part 1:

- Hand out: 18th November
- Hand in: 20th December, 17h

Part 2:

- Hand out: 12th December
- Hand in: 17th January, 17h

- The Exam will cover the lectures' material and additional material provided on Brightspace

Course Information - Assignments

- Submit your **report** and **model script file(s)** through BrightSpace as individual files. Include group number, names, and student IDs in these files. Files submitted by email will not be considered.
- If you fail to meet the **deadline**, 0.5 points will be deducted from your grade for each day after the deadline.
- If you fail to obtain a **grade of 5.5** or higher you will fail the assignment. In that case, you will get a chance to improve your work and pass the assignment. Your final grade cannot then become higher than 5.5.
- Include an overview of the **workload distribution** of each group member. Indicate (in percentages) each member's contribution to the three categories: mathematical modelling (30%), programming (50%), and reporting (20%). You will receive an individualized grade for the assignment.

Course Information - Workload

AE4423 ⇒ 4 ECTS (112 working hours)

Activity	Working Hours
Lectures	20
Assignment including 6 studio lectures (80×3 students = 240 hrs/group = 30 days/group)	80
Exam preparation & Exam	12
TOTAL	112

Outline - Lecture 1

① Course Information

- ① Course Information
 - a Overview
 - b Content
 - c Staff
 - d Learning Goals
 - e Lectures
 - f Materials
 - g Grading
 - h Workload

② Airline Planning Framework

③ Demand Analysis

- ③ Demand Analysis
 - a Dichotomy
 - b Mismatch
 - c Spoilage & Spillage

④ Demand Models

- ④ Demand Models
 - a Time-Series
 - b Causal Methods
 - c Summary

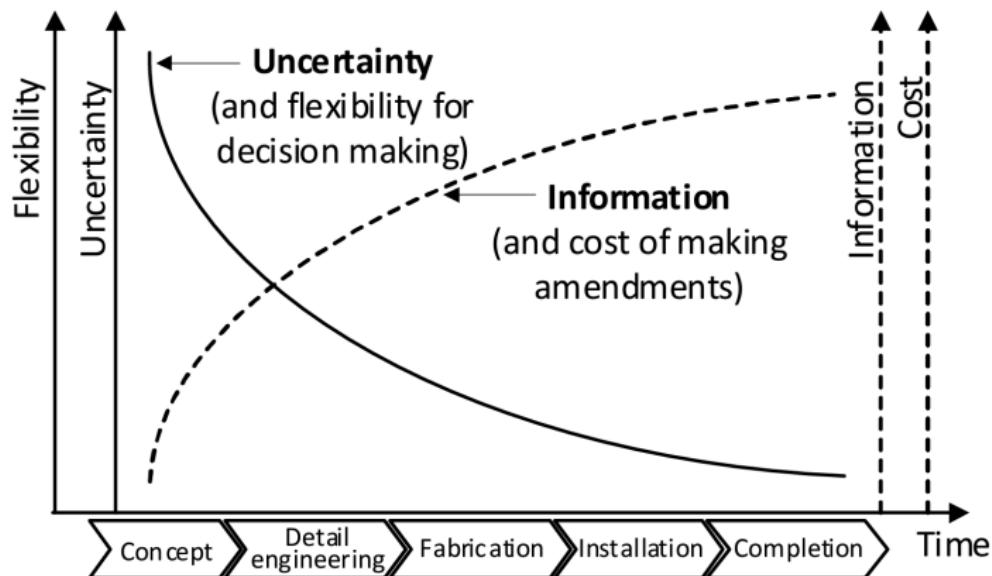
⑤ Market Share

- ⑤ Market Share
 - a S-Curve Model
 - b Quality of Service Index
 - c Logit Model

⑥ Homework

Airline Planning Framework

Uncertainty vs Time Before Departure



Source: Santhiapillai, F.P. & Ratnayake, R.M.. (2018). Identifying and Defining Knowledge-work Waste in Product Development: A Case Study on Lean Maturity Assessment. 834-838. 10.1109/IEEM.2018.8607682

Airline Planning Framework (2)



Strategic Decisions (10 - 1 year in advance):

- Definition of the strategy (vision)
- Match resources (supply) with forecasted demand

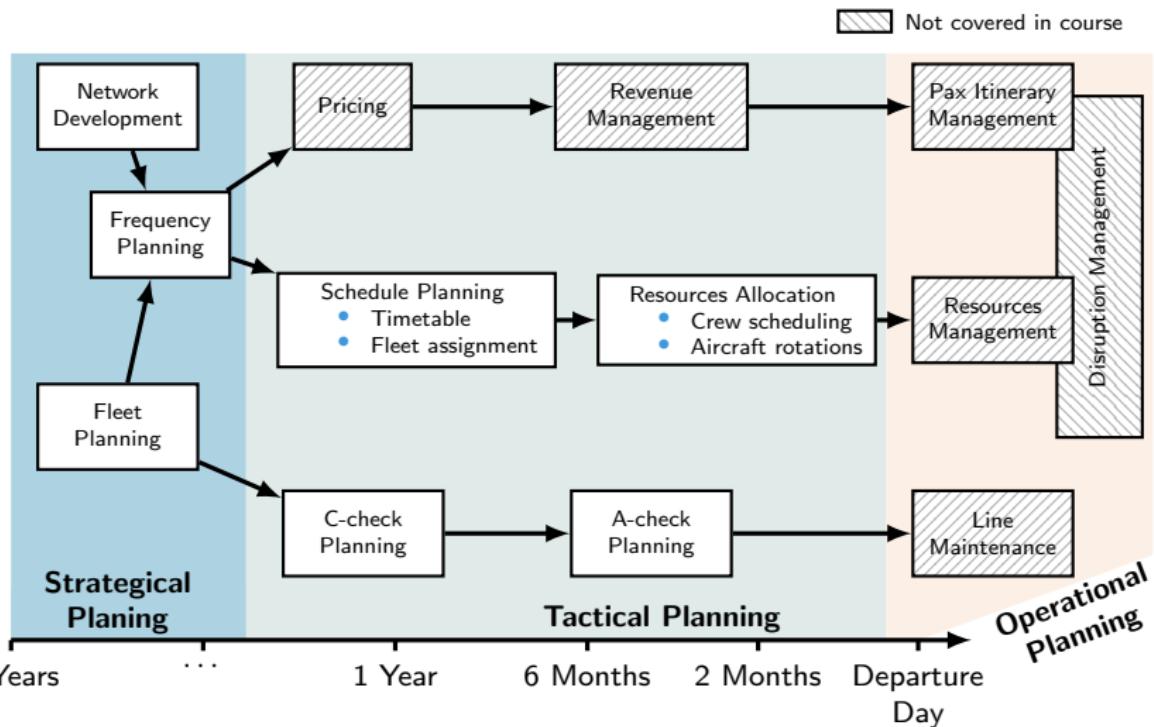
Tactical Decisions (1 year - 1 month):

- Readjust to the market situation
- Maximize the potential revenue generated by resources

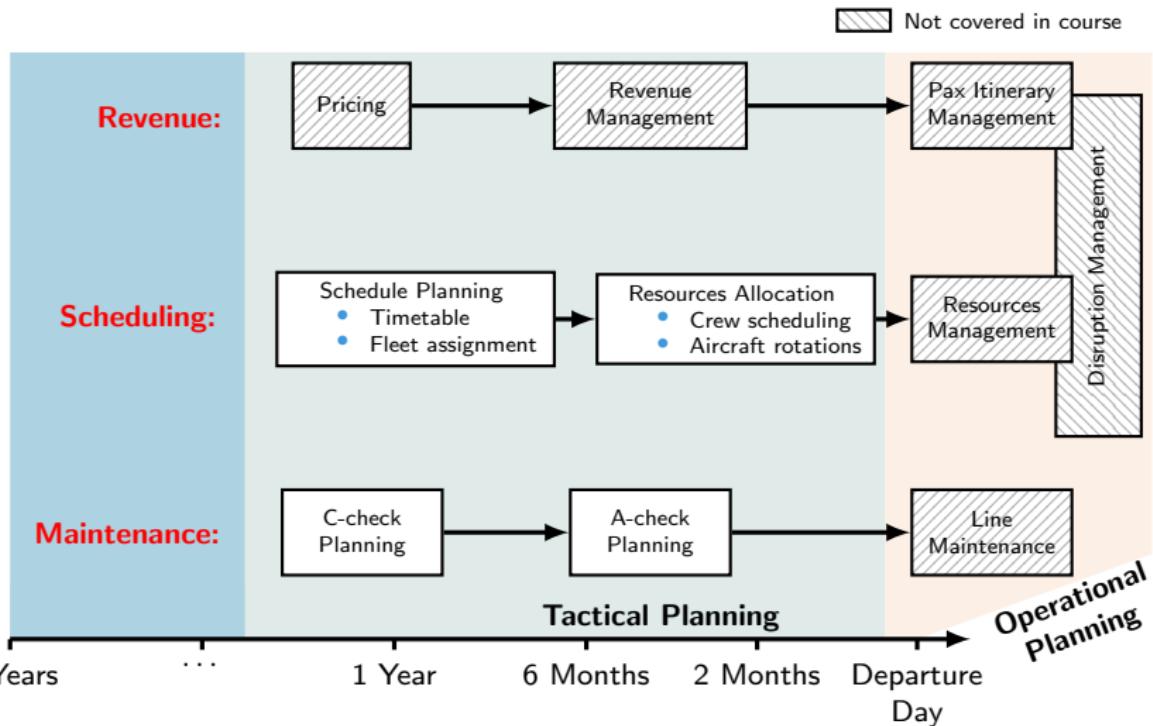
Operational Decisions (4 weeks - 1 day before the day of operations):

- Control daily operation – mitigate additional costs

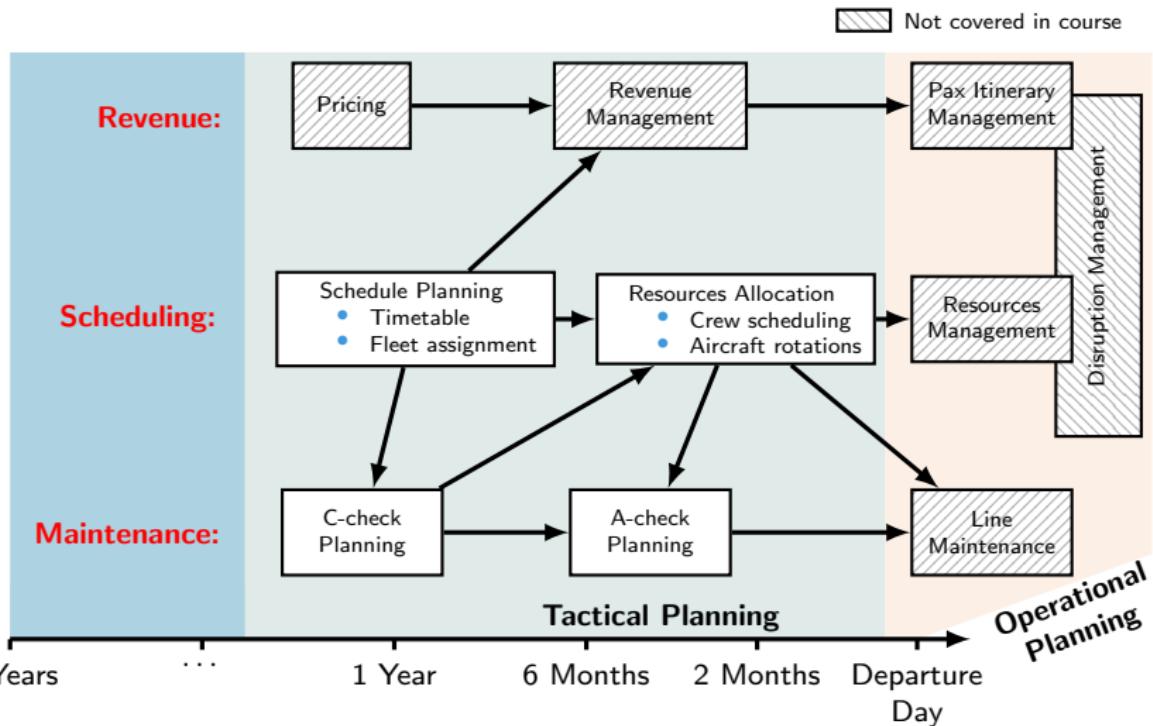
Airline Planning Framework (3)



Airline Planning Framework (4)



Airline Planning Framework (5)



Outline - Lecture 1

① Course Information

- ① Course Information
 - a Overview
 - b Content
 - c Staff
 - d Learning Goals
 - e Lectures
 - f Materials
 - g Grading
 - h Workload

② Airline Planning Framework

③ Demand Analysis

- ③ Demand Analysis
 - a Dichotomy
 - b Mismatch
 - c Spoilage & Spillage

④ Demand Models

- ④ Demand Models
 - a Time-Series
 - b Causal Methods
 - c Summary

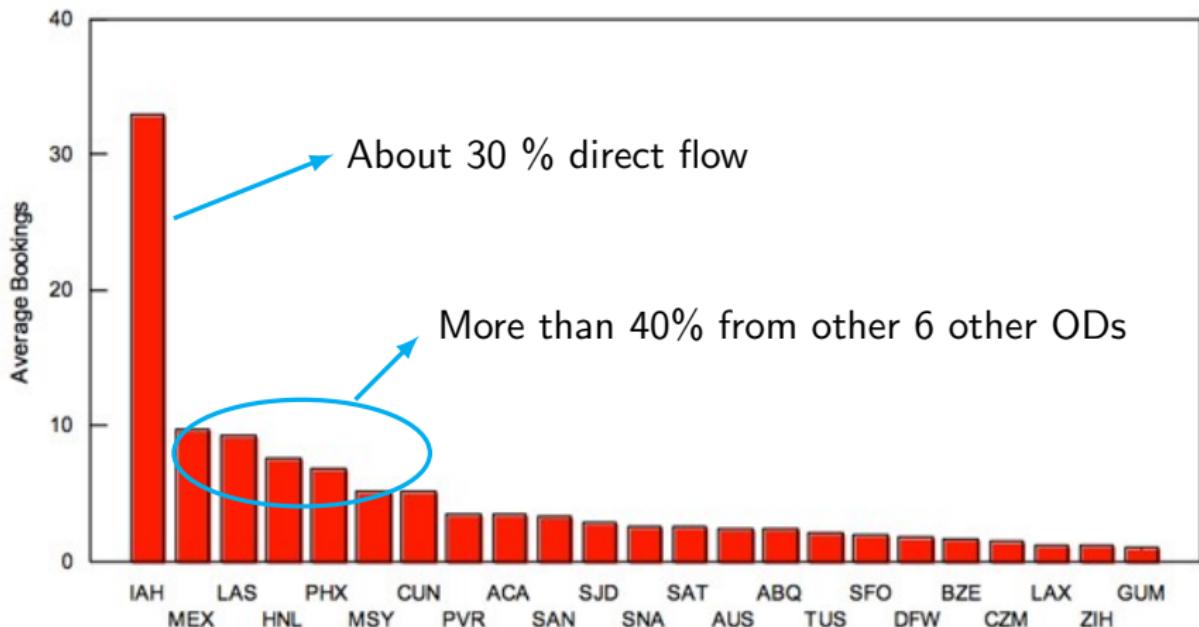
⑤ Market Share

- ⑤ Market Share
 - a S-Curve Model
 - b Quality of Service Index
 - c Logit Model

⑥ Homework

Demand & Supply

Joint supply of service to multiple O-D markets (BOS-IAH Flight)



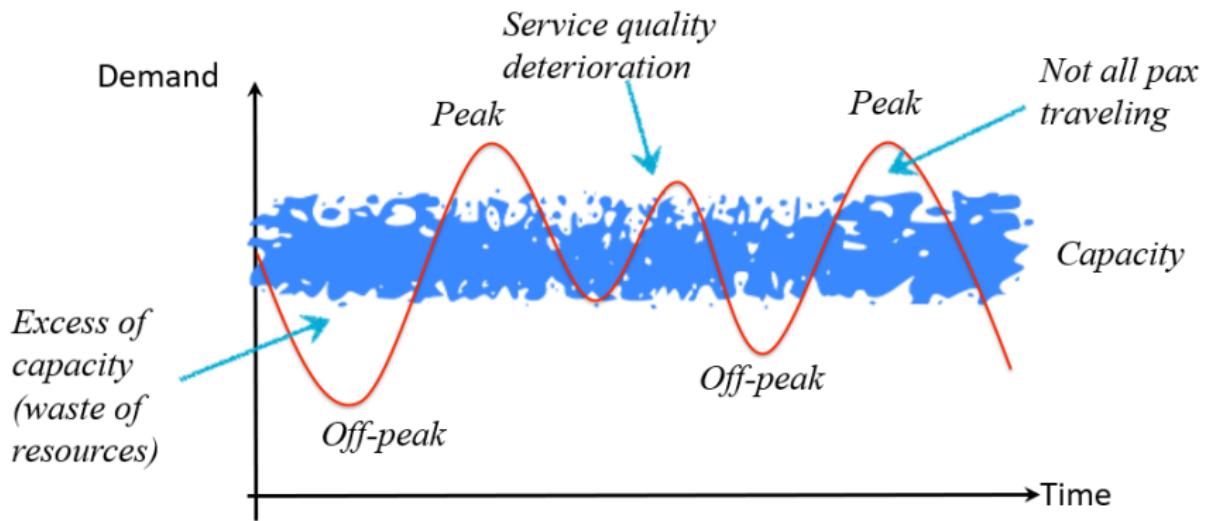
Source: P. Belobaba - MIT Lectures (16.75J/1.234J), 2006

Demand & Supply - Dichotomy

- Air travel demand is defined for an **origin-destination (OD)** market, but supply is offered by **flight leg**
- There is an inherent inability to directly **compare demand and supply** in an individual market
- It is hard, for instance, to know if:
 - ① A market is in 'equilibrium'
 - ② The airline is profitable in a specific flight leg
- These questions can only be answered at the level of an entire network of flights.

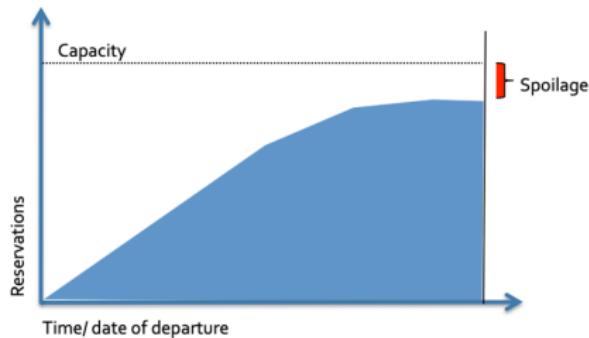
Demand & Supply - Mismatch

In air transport, the **supply** rarely matches with the **demand**:

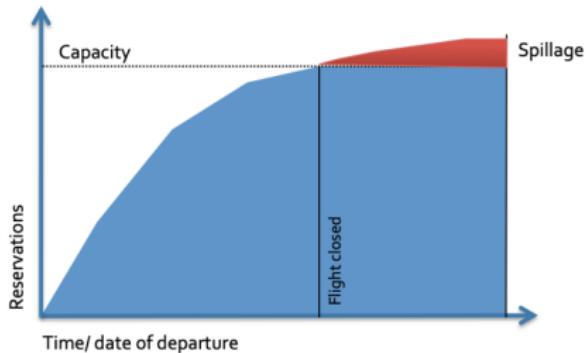


Spoilage & Spillage

Spoilage: not fulfilling the entire capacity of the aircraft, given the low demand compared to capacity offered.

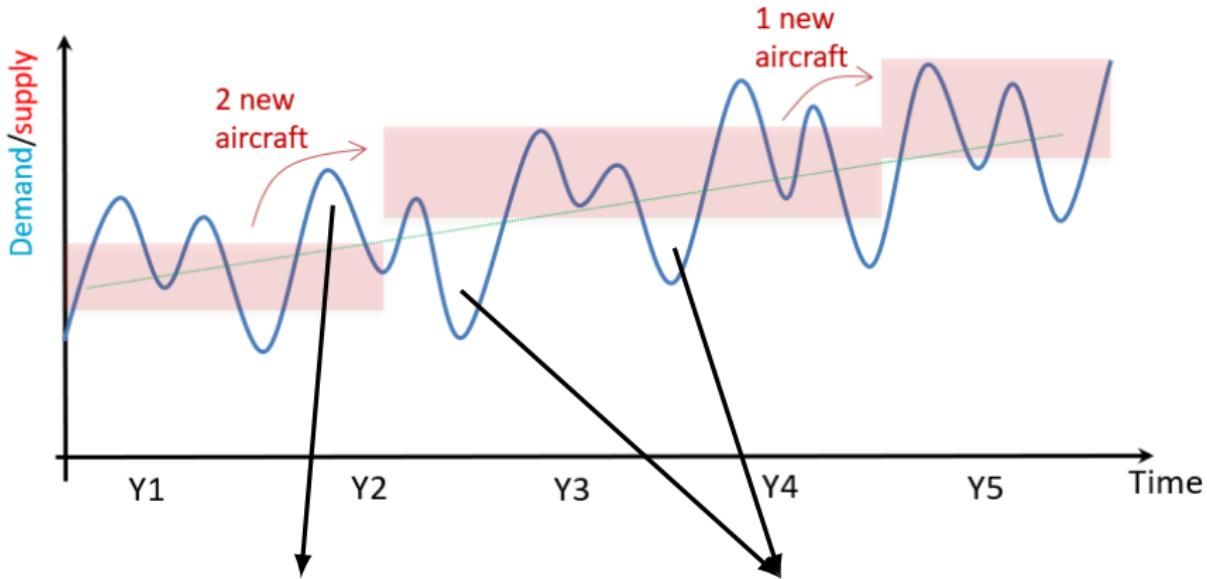


Spillage: losing demand when not enough capacity is provided. Some potential passengers are 'lost'.



Demand & Supply - Mismatch (2)

Fleet composition and utilization to cope with demand variations:



How do airlines solve this lack of capacity?

What do airlines with this surplus of capacity?

Outline - Lecture 1

① Course Information

- ① Course Information
 - a Overview
 - b Content
 - c Staff
 - d Learning Goals
 - e Lectures
 - f Materials
 - g Grading
 - h Workload

② Airline Planning Framework

③ Demand Analysis

- ③ Demand Analysis
 - a Dichotomy
 - b Mismatch
 - c Spoilage & Spillage

④ Demand Models

- ④ Demand Models
 - a Time-Series
 - b Causal Methods
 - c Summary

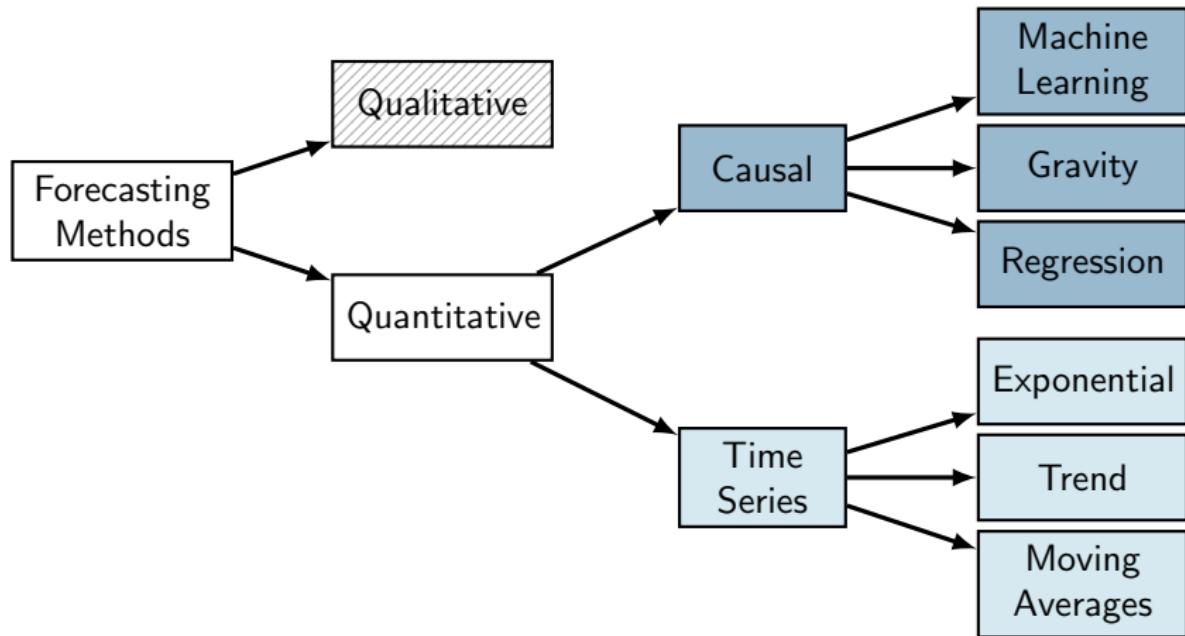
⑤ Market Share

- ⑤ Market Share
 - a S-Curve Model
 - b Quality of Service Index
 - c Logit Model

⑥ Homework

Demand Models

The existing demand models can be divided in:



Time-Series Projections

Technique most used by airlines. Establish the relationship between **traffic** (the dependent variable) and **time** (the independent variable).

Involves a projection into the future of what has happened in the past.

Assumes that the factors affected demand in the past will continue to operate in the same manner in the future.

- Simple (linear) trend:

$$D_t = D_0 + \text{var} \times t,$$

where D_0 is the 'y-interception' value and var is the constant time growth (e.g., annual growth).

- Average rate of growth or exponential trend:

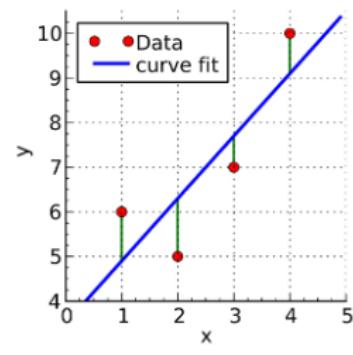
$$D_t = D_0(1 + \text{AAG})^t,$$

where D_t is the forecasted demand for time t , D_0 is the 'y-interception' value, and AAG is the average (e.g., annual) demand growth.

Time-Series - Calibration

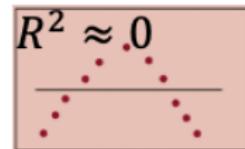
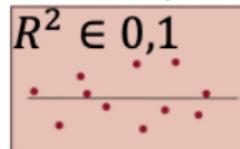
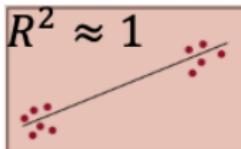
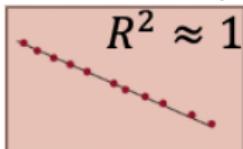
- Calibration - done using Ordinary Least Squares (OLS).

Idea: To minimize the residual values. That is, to minimize the distances between data and the curve fit:



$$\min_{\beta} \sum_{i=1}^N (y_i - \beta x_i)^2$$

- Goodness of fit (coefficient of determination R^2)



Time-Series - Autoregressive Moving Average (ARMA)

A linear combination of models Autoregressive with a Moving-average:

$$D_t = c_a + \sum_{i=1}^p \alpha_i D_{t-1} + \varepsilon_t \quad D_t = c_m + \varepsilon_t + \sum_{j=1}^m \beta_j \varepsilon_{t-1}$$

$$D_t = c + \varepsilon_t + \sum_{i=1}^p \alpha_i D_{t-1} + \sum_{j=1}^m \beta_j \varepsilon_{t-1}$$

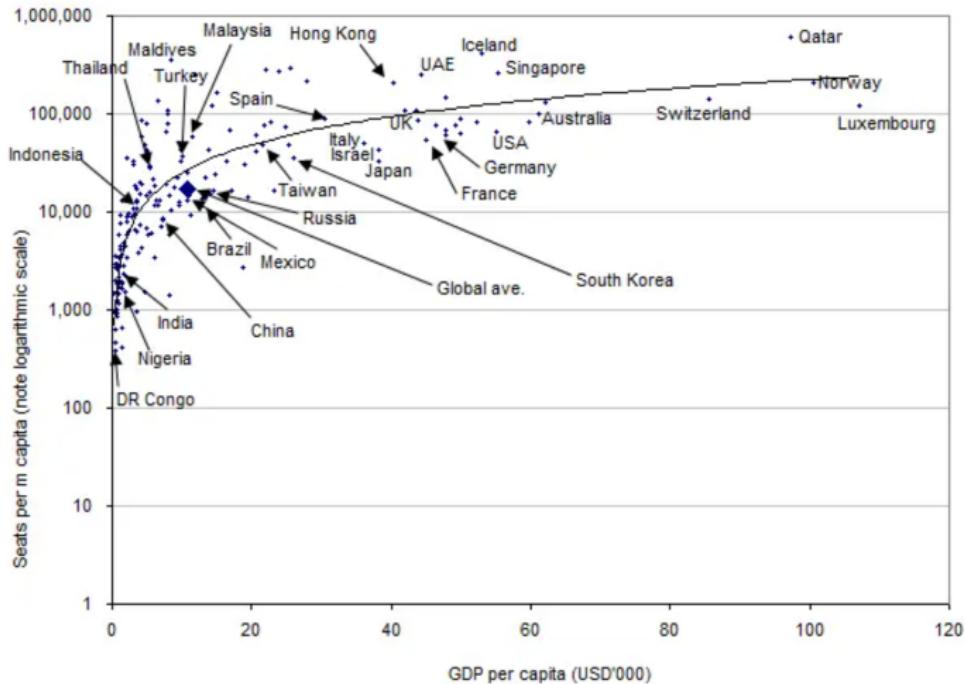
where:

- D_t is the forecasted demand for time t
- c is a **calibration** constant
- ε_t is (white) noise with $E(\varepsilon_t) = 0$ and variance σ^2
- α_i and β_j are calibration parameters

Note: The values of the parameters are usually computed using Ordinary Least Square or maximum likelihood estimations.

Watch ritvikmath

Causal Methods



Source: CAPA - Centre for Aviation, OAG (seat data for week of 9-Jun-2014), International Monetary Fund

Causal Methods (2)

General Methodology

- ① **Identify** and select the factors (independent or explanatory variables).
- ② **Determine** the functional relationship between the factors and the level of demand (dependent variable) - i.e., specify the form of the model to be used and test correlations.
- ③ **Calibrate** the model, calculating the parameters of the model that better fit to the past data.
- ④ **Analyse** the resulting relationship to check if the significance of each factor, the market logic, and the statically robustness of the model.
- ⑤ **Validate** the results comparing with extra data or with other sources forecast.

Causal Methods (3)

Demand depends on multiple **factors**:

- Trip fare
- Travel time
 - Number of connections
 - Frequency
- Service
 - In-flight and at airport
 - Boarding and check-in rules
- Economic factors (e.g., GDP)
- Fuel prices
- Socio-economic factors
- Political drivers
- Travel purpose
- Passenger characteristics
 - Age
 - Activity and Income
 - Household
- ...

Causal Methods - Regression Models

General form for regression models (or econometric models):

$$D = f(P, Y, F, T, \dots)$$

One possible formula is

$$D = K \times P^a \times Y^b \times F^c \times T^d$$

or

$$D = K + a \times P + b \times Y + c \times F + d \times T$$

where:

- where D is the forecasted demand
- K is a **calibration** constant representing the market size
- P is a demographic variable (e.g., population)
- Y is an economic value (e.g., GDP, income per capita)
- F is the average price of air travel
- T is total trip time, reflecting frequency and flight time
- a, b, c, d are model **calibration** parameters (and, in the first formula, represent the elasticity of demand to each factor)

Note: The values of the parameters are usually computed using Ordinary Least Square or maximum likelihood estimations.

Causal Methods - Regression Models - Gravity Model

It is a particular regression model formulation:

- Widely used in long-haul transport demand forecast
- Suitable for new routes, for each historic data is rare or nonexistent

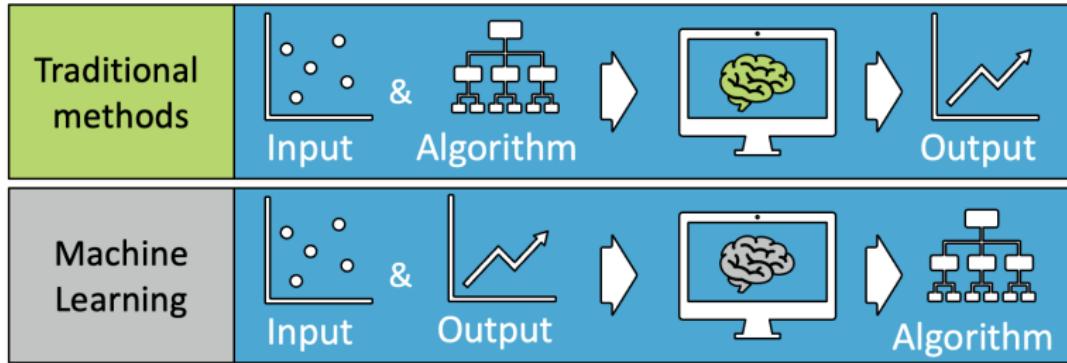
$$D_{ij} = K \frac{P_i P_j}{T_{ij}^\beta}$$

where:

- D_{ij} is the forecasted demand between i and j
- K is a **calibration** parameter representing the market size
- P_i is the dimension (mass) of demand node i (e.g., population, GDP, total traffic at the airport)
- T_{ij} is a measure of distance between i and j (total travel time, travel price, or a combination)
- β is a **calibration** parameter, representing the distance impedance factor

Note: The values of the parameters are usually computed using Ordinary Least Square

Causal Methods - Machine Learning



Main machine-learning regression methods:

- Ridge Regression
- Random Forests (with regression trees)
- Gradient Boosting (ensemble of learning algorithms)
- Neural Networks

Causal Methods - Machine Learning - Ridge Regression

Used when:

- Data suffers from **multicollinearity**
- **Few data samples** (< number of features)

A regression analysis with a penalised cost 'function':

$$Y_i = \sum_{j=1}^p X_{ij} \times \beta_j + \varepsilon_i$$

subject to:

$$\min_{\beta_j} \underbrace{\sum_{i=1}^N \left(y_i - \sum_{j=1}^p X_{ij} \times \beta_j \right)^2}_{\text{Residuals} \quad (= \text{linear regression})} + \underbrace{\lambda}_{\text{Tuning parameter} \quad (\text{pre-chosen constant})} \times \underbrace{\sum_{j=1}^p \beta_j^2}_{\text{Regularization estimator} \quad (\text{can assume different forms})}$$

Note: the penalty makes the values from β_j be preferably small.

Causal Methods - Machine Learning - Random Forests

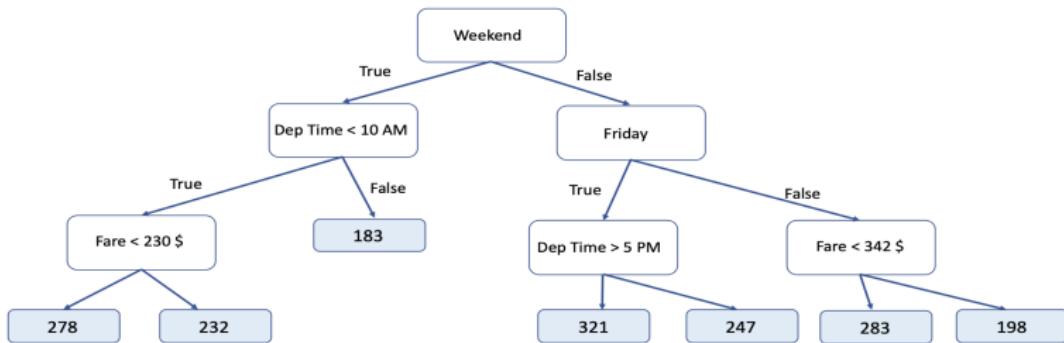
Composed by a ‘forest’ of decision trees. It builds multiple decision trees and merges them together to get a more accurate and stable prediction. It merges (for regression) the average value from the multiple trees

Useful:

- When no linear dependence exists
- Data suffers from **multicollinearity**
- When missing values in data samples
- Dealing with a **large data set**
- To estimate relative importance of each input feature

To avoid overfitting, it should have several parallel trees with:

- Different subsets of data drawn at random (bootstrap)
- Random selection of features (feature bagging)



Watch StatQuest

Causal Methods - Machine Learning - Gradient Boosting

Ensemble learning technique that combines the predictions of multiple weak learners (usually decision trees) to create a robust predictive model.

There are several types of Gradient Boosting, each with unique characteristics:

- XGBoost (Extreme Gradient Boosting)
- LightGBM (Light Gradient Boosting Machine)
- CatBoost (Categorical Boosting)

Advantages:

- High Accuracy: often outperforms other algorithms
- Handles Mixed Data Types: both numerical or categorical
- Feature Importance: provides insights into which features are most influential

Disadvantages:

- Computationally Intensive: training can be slow
- Hyperparameter Tuning: finding best hyperparameters is time-consuming
- Potential Overfitting: without proper tuning, it's prone to overfitting the training data

Causal Methods - Machine Neural Networks

Analogy to how biological organisms process information. Biological brains contain neurons connected by synapses that facilitate information transfer between neurons.

Is a variational non-linear function that maps input data to a desired output. The connections in the graphical representation means that the output from one set of neurons serves as the input for the next set of neurons.

- Remarkably powerful - under structure assumptions can approximate any smooth function arbitrarily well as the number of neurons tends to infinity.
- Typically depends on a large amount of parameters.

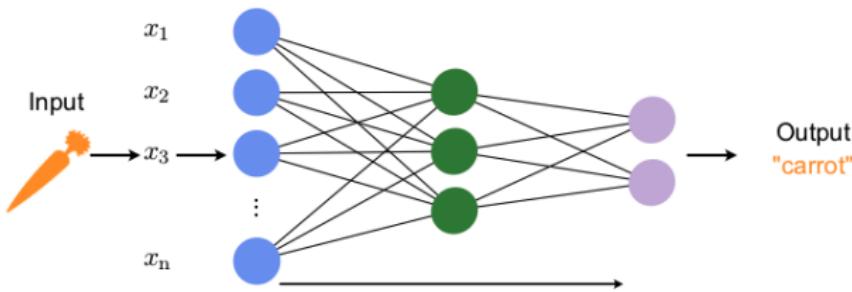


Image Source

Demand Forecast Models - Summary

	Qualitative Methods		Time-series Projections			Causal Methods		
	Executive Judgment	Market Research	Exponential Trend	Linear Trend	ARMA	Regression	Gravity Model	Machine Learning
Accuracy	Short-term (0-6m)	++	++	++/+	++/+	++	++	++
	Long-term (>6yrs)	-	+/-	-	-	+/-	+	+
Good for forecasting?	Growth	+	++	++	++	++	++	++
	Reaction	+	++				-	++
	Unstable	-	+			+	-	++
	New routes	-	+			+	++	+
Can find turning point?		+/-	+	-	-	+/-	++	-
Amount of data required [days]		1-2	90+	1-2	1-2	1-2	30-90	20-0
Cost of implementation		++	--	+	+	+	-	--

In a very general way:

Accuracy 

vs

Amount of data
Cost of implementation 

Outline - Lecture 1

① Course Information

- ① Course Information
 - a Overview
 - b Content
 - c Staff
 - d Learning Goals
 - e Lectures
 - f Materials
 - g Grading
 - h Workload

② Airline Planning Framework

③ Demand Analysis

- ③ Demand Analysis
 - a Dichotomy
 - b Mismatch
 - c Spoilage & Spillage

④ Demand Models

- ④ Demand Models
 - a Time-Series
 - b Causal Methods
 - c Summary

⑤ Market Share

- ⑤ Market Share
 - a S-Curve Model
 - b Quality of Service Index
 - c Logit Model

⑥ Homework

Market Share

Airlines **compete for passengers and market share** based on the following factors:

- Frequency of flights and departures schedule
- Price charged, relative to other airlines
- Quality of service and products offered, including airport and in-flight service amenities and/or restrictions on discount fare products

Passengers choose combination of flight schedules, prices, and product quality that minimizes disutility of air travel.

Rule of Thumb

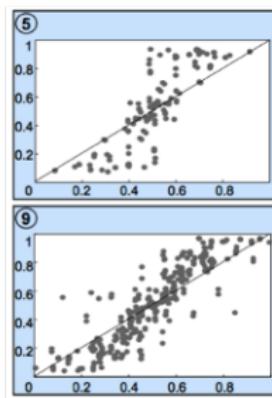
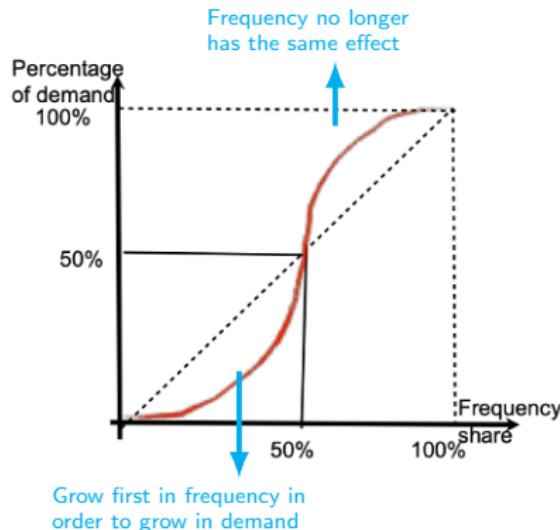
With all else equal, **airline market shares** will approximately equal their **frequency shares**. But there is empirical evidence of an **S-curve** relationship.

Market Share - S-Curve Model

The mathematical model formulation of this S-curve relationship:

$$MS(i) = \frac{FS(i)^a}{FS(A)^a + FS(B)^a + FS(C)^a + \dots}$$

where $MS(i)$ is the market share of airline i , $FS(i)$ the non-stop frequency share of airline i , and a an exponent greater than 1.0 (generally between 1.3 - 1.7) defining the shape of the S-Curve.



Source: IATA, SEPTEMBER 2006

Market Share - S-Curve Model (2)

Frequency is not necessarily the only factor! The S-curve model can be extended to include many other decision factors.

The 'quality of service index' or **QSI model** assigns a score to each carrier based on a range of factors, such as:

- Frequency
- Quality of service
- Directness of service
- Travel time and fare
- Aircraft type
- Scheduling convenience
- Carrier preference

Market Share - Quality of Service Index (QSI)

The mathematical formulation of this QSI model is:

$$MS(i) = \frac{Index(i)}{Index(A) + Index(B) + Index(C) + \dots}$$

with (example):

$$Index(i) = FS(i)^a \times Fare(i)^b \times Capacity(i)^c \times Connection(i)^d \times Aircraft(i)^e \times \dots$$

where:

- $MS(i)$ is the market share of airline i
- $FS(i)$ is the frequency share of the airline in that flight leg
- $Fare(i)$ is the average ticket value from the airline in that flight leg
- $Capacity(i)$ is the amount of seats offered by the airline in that flight leg
- $Connection(i)$ is the connection type (direct or connecting)
- $Aircraft(i)$ is the type of aircraft (e.g, jet, propeller, new, old)
- a, b, c, \dots model parameters that need to be calibrated with past data

Market Share - Logit Model

These are discrete choice (or 'logit') models that describe how **individuals chose one alternative among a finite set of alternatives.**

It follows the maximum utility concept, in which **utility** for an individual is modelled as:

$$V_{ni} = \beta_{x_{ni}} + \varepsilon_{ni}$$

where:

- V_{ni} : total observed utility of alternative i for individual n
- β : vector of estimated parameters associated with attributes x
- x_{ni} : vector of attributes for alternative i for individual n
- ε_{ni} : unobserved error component

The values of the parameters can be computed using maximum likelihood estimations.

Market Share - Logit Model (2)

- Assuming that the error term is independent and identically distributed Gumbel (i.e., Type I extreme value), we have the widely used multinomial logit model (MNL) (McFadden 1974).
- The MNL probability of selecting alternative i among all j alternatives in C_n , the choice set of individual n , is expressed as:

$$P_{ni} = P(i|V_{ni}, \beta) = \frac{e^{\beta V_{ni}}}{\sum_j e^{\beta V_{nj}}}$$

- The demand for a given alternative i can then be estimated by multiplying the total demand (D) by the probability of selecting alternative for each individual:

$$d_i = \sum_n P_{ni} D_n$$

Outline - Lecture 1

① Course Information

- ① Course Information
 - a Overview
 - b Content
 - c Staff
 - d Learning Goals
 - e Lectures
 - f Materials
 - g Grading
 - h Workload

② Airline Planning Framework

③ Demand Analysis

- ③ Demand Analysis
 - a Dichotomy
 - b Mismatch
 - c Spoilage & Spillage

④ Demand Models

- ④ Demand Models
 - a Time-Series
 - b Causal Methods
 - c Summary

⑤ Market Share

- ⑤ Market Share
 - a S-Curve Model
 - b Quality of Service Index
 - c Logit Model

⑥ Homework

Homework - Reading Material

On Brightspace, you can find the following reading material:

- Airlines' Key Performance Indicators (KPIs)
- KPIs Calculation
- Air Service Agreements (ASAs) & Freedoms

These documents contain content for the exam & assignments.

Airline Planning and Optimisation - AE4423-20

Lecture 2 Multi-commodity Flow Problems

Marta Ribeiro
Assistant Professor
Air Transport & Operation
Control & Operations

Delft University of Technology, The Netherlands

November, 2024

Course Information - Content

Management

Planning Framework
Performance Indicators
Demand and Supply
Market Share

Tactical Planning

Passenger Mix Flow
Fleet Assignment
Aircraft Rotation
Crew Scheduling
Maintenance



Strategical Planing

Network Structure
Demand Forecast
Network Planning
Fleet Planning

Operations Research

MILP Models
Multi-commodity Flow Problems
Shortest-path Algorithms
Column Generation Algorithm
Dynamic Programming

Outline - Lecture 2

① Multi-commodity Flow Models

- ① a Decision Variables & Parameters
- ① b Objective Function
- ① c Constraints
- ① d Problem Formulation
- ① e Example

② Mixed Integer Linear Programming

- ② a Python & Gurobi
- ② b Results
- ② c Output
- ② d Tableau
- ② e Simplex
- ② f Simplex & Dual Theory

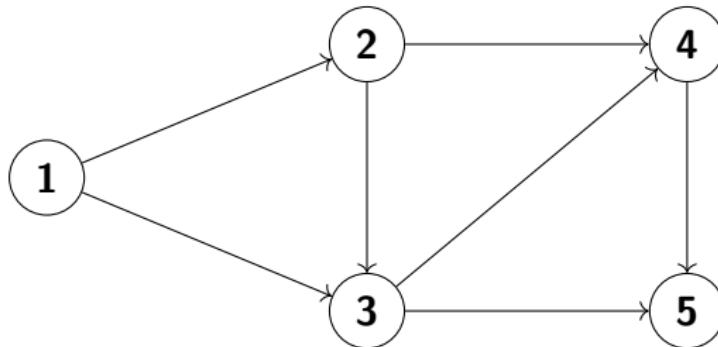
③ Homework

Multi-commodity Flow Models (MCFM)

Multi-commodity flow problem: **network flow problem** with multiple commodities (e.g., cargo items, passengers per class, vehicle types) between different **source** and **sink** nodes.

Find the **minimum cost** flow of a set of commodities through a network, where the arcs have an individual capacity for each commodity, and a mutual capacity for all the commodities.

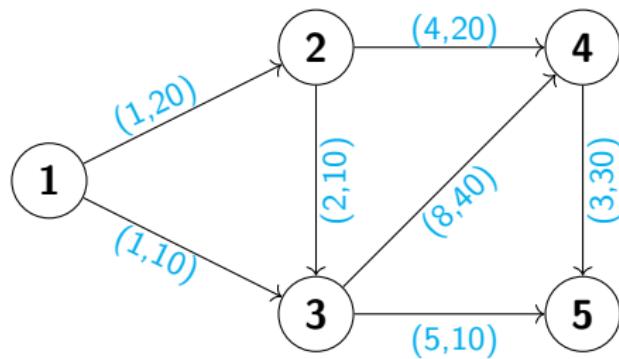
In the airline industry, the multi-commodity model is adopted to formulate **crew pairing** and **fleet assignment** models.



MCFM - Example

What are the flows in the network that minimise costs and transport all commodities, given the 4 commodities and the capacity in the arcs?

Note: **Arcs (cost, capacity)**



Commodities		
From (i)	To (j)	Quantity (k)
1	4	15
1	5	5
2	5	10
3	5	5

MCFM - Decision Variables & Parameters

Decision variables (choice variables) examples:

- **Flows per arc:**
 - Passenger flows (per class)
 - Frequency per route
- Number of AC to buy
- Aircraft flying the route or not (binary)

Parameters (input values, constants, or dependent variables) examples:

- **Arc costs:**
 - Costs of flying (airline or pax)
 - Fares (revenue or pax's cost)
- Number of AC in the fleet
- **Capacity per arc**
- **Demand values**

MCFM - Objective Function

The **objective function** (OF) – the goal of the problem (minimization or maximization function). Examples:

Sets:

K : Set of commodities

N : Set of nodes

L : Set of arcs

- Min costs associated with flow:

$$\min C = \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} c_{ij}^k \times pax_{ij}^k$$

- Max revenue:

$$\max R = \sum_{i \in N} \sum_{j \in N} yield_{ij} \times pax_{ij} \times dist_{ij}$$

- Max demand served:

$$\max D = \sum_{i \in L} pax_i$$

- Min spillage:

$$\min S = \sum_{i \in N} \sum_{j \in N} (Dem_{ij} - pax_{ij})$$

MCFM - Constraints

The **constraints** are conditions for the variables that are required to be satisfied.

Examples:

Sets:

K : Set of commodities

N : Set of nodes

L : Set of arcs

- Flow continuity at the node (per class):

$$\sum_{j \in N} pax_{ij}^k - \sum_{j \in N} pax_{ji}^k = Dem_i^k \quad \forall i \in N, k \in K$$

- Demand constraint:

$$\sum_{j \in N} pax_{ij}^k = Dem_i^k \quad \forall i \in N, k \in K$$

- Capacity constraint:

$$\sum_{k \in K} pax_{ij}^k \leq Cap_{ij} \quad \forall i, j \in N$$

- Budget constraint:

$$\sum_{n \in AC} Cost_n \times AC_n \leq Budget$$

MCFM - Problem Formulation

Sets: K : Set of commodities

N : Set of nodes

L : Set of arcs

A general multi-commodity problem can be formulated as follows:

$$\min C = \sum_{k \in K} \sum_{m \in L} c_m^k \times x_m^k$$

Subject to:

$$\begin{aligned} \overbrace{\sum_{m \in L_i^{out}} x_m^k}^{\text{Outflow}} - \overbrace{\sum_{m \in L_i^{in}} x_m^k}^{\text{Inflow}} &= Dem_i^k && \text{if } i \in O(K) \\ &= -Dem_i^k && \text{if } i \in D(K) \\ &= 0 && \text{otherwise} \end{aligned} \quad \forall i \in N, k \in K$$

$$\sum_{k \in K} x_{ij}^k \leq Cap_m \quad \forall m \in L$$

$$x_m^k \in \mathbb{Z}_0^+$$

MCFM - Problem Formulation (2)

Sets: K : Set of commodities

N : Set of nodes

L : Set of arcs

A general multi-commodity problem can be formulated as follows:

$$\min C = \sum_{k \in K} \sum_{m \in L} c_m^k \times x_m^k$$

Arcs Costs DV: Flows

Subject to:

$$\begin{aligned} \underbrace{\sum_{m \in L_i^{out}} x_m^k}_{\substack{\text{Outflow} \\ \text{Arcs starting at } i}} - \underbrace{\sum_{m \in L_i^{in}} x_m^k}_{\substack{\text{Inflow} \\ \text{Arcs ending at } i}} &= Dem_i^k && \text{Demand for } k \\ &&& \text{if } i \in O(K) \\ &= -Dem_i^k && \text{if } i \in D(K) \\ &= 0 && \text{otherwise} \end{aligned}$$

Origin of k Destination of k $\forall i \in N, k \in K$

Nodes Commodities

$$\sum_{k \in K} x_{ij}^k \leq Cap_m \quad \forall m \in L$$

Capacity in m

$$x_m^k \in \mathbb{Z}_0^+$$

Capacity Constraint

MCFM - Problem Formulation (3)

Create loop cycles for every node, commodity!

A general multi-commodity problem can be formulated as follows:

$$\min C = \sum_{k \in K} \sum_{m \in L} c_m^k \times x_m^k$$

Subject to:

$$\underbrace{\sum_{m \in L_i^{out}} x_m^k}_{Outflow} - \underbrace{\sum_{m \in L_i^{in}} x_m^k}_{Inflow} = Dem_i^k \quad \text{if } i \in O(K)$$
$$= -Dem_i^k \quad \text{if } i \in D(K)$$
$$= 0 \quad \text{otherwise}$$

$\forall i \in N, k \in K$

$$\sum_{k \in K} x_{ij}^k \leq Cap_m \quad \forall m \in L$$

$$x_m^k \in \mathbb{Z}_0^+$$

Outline - Lecture 2

1 Multi-commodity Flow Models

- a Decision Variables & Parameters
- b Objective Function
- c Constraints
- d Problem Formulation
- e Example

2 Mixed Integer Linear Programming

- a Python & Gurobi
- b Results
- c Output
- d Tableau
- e Simplex
- f Simplex & Dual Theory

3 Homework

Mixed Integer Linear Programming (MILP)

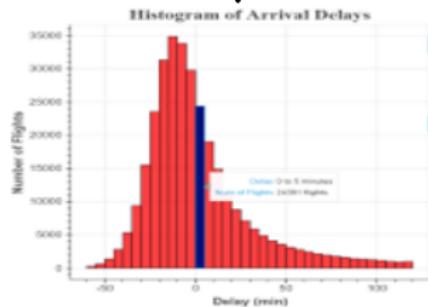
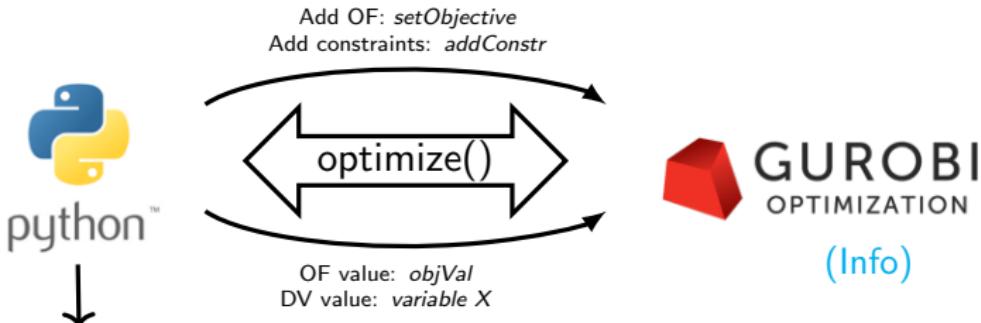
Linear programming (LP) is a method to achieve the best outcome (e.g., maximum profit, lowest cost) in a mathematical model whose requirements and objective are represented by linear relationships.

Mixed Integer LP (MILP) is an LP in which some of the variables are integer.

Find solution: x and y
that maximizes/minimizes: $c^T x + d^T y$
subject to:
 $Ax \leq b$
 $Ey \leq f$
 $A \in \mathbb{R}^{m \times n}, E \in \mathbb{R}^{p \times q}$
 $b \in \mathbb{R}^m, f \in \mathbb{R}^p, c \in \mathbb{R}^n, d \in \mathbb{R}^q$
 $x \in \mathbb{Z}^n, y \in \mathbb{R}^q$

MILP - Python & Gurobi

Framework used to solve the optimisation problem:



Results Analysis

MILP - Python & Gurobi - Variables

```
#=====
# Create the model which represent the nodes, arcs, and commodities
#=====

class Arc:
    def __init__(self, origin, destination, cost, capacity):
        self.From = origin      #i
        self.To = destination #j
        self.Cost = cost
        self.Capac = capacity

class Commodity:
    def __init__(self, origin, destination, quantity):
        self.From = origin      #i
        self.To = destination #j
        self.Quant = quantity

class Node:
    def __init__(self, id):
        self.id = id
        self.InLinks = [ ]
        self.OutLinks = [ ]

    def addInLink(self, Node):      # Add new 'In Link' to the node
        self.InLinks.append(Node)

    def addOutLink(self, Node):     # Add new 'Out Link' to the node
        self.OutLinks.append(Node)
```

Arc Features

Commodities
Features

Nodes
Features

MILP - Python & Gurobi - Objective Function

$$\min \quad C = \sum_{k \in K} \sum_{m \in L} c_m^k \times x_m^k$$

```
# LP model (this is an object)
model = Model("MCF")

# Decision Variables
# For addVar check https://www.gurobi.com/documentation/current/refman/py\_model\_addvar.html
# vtype = 'C' for continuous
x = {}
for Arc in Arcs:
    for k in range(len(Commodities)):
        x[k,Arc.From,Arc.To] = model.addVar(obj=Arc.Cost, vtype ='C',
                                              name = ''.join(['Arc(', str(Arc.From), ',', str(Arc.To), ')'])))

# update model with the DVs before adding constraints
model.update()
```

MILP - Python & Gurobi - Objective Function (2)

$$\min C = \sum_{k \in K} \sum_{m \in L} c_m^k \times x_m^k$$

Cost coefficients multiplying by x in the Objective Function

```
# LP model (this is an object)
model = Model("MCF")

# Decision Variables
# For addVar check https://www.gurobi.com/documentation/current/refman/py\_model\_addvar.html
# vtype = 'C' for continuous
x = {}
for Arc in Arcs:
    for k in range(len(Commodities)):
        x[k,Arc.From,Arc.To] = model.addVar(obj=Arc.Cost, vtype ='C',
                                              name = ''.join(['Arc(', str(Arc.From), ',', str(Arc.To), ')'])))

# update model with the DVs before adding constraints
model.update()
```

MILP - Python & Gurobi - Constraints

$$\sum_{m \in L_i^{\text{out}}} x_m^k - \sum_{m \in L_i^{\text{in}}} x_m^k$$

```
# build 'balance' constraints
# for .addConstr() check https://www.gurobi.com/documentation/current/refman/py\_model\_addconstr.html
Balance = {}
for k in range(len(Commodities)):
    for From in range(len(Nodes)):
        balance = quicksum(x[k,From>To] for To in Nodes[From].OutLinks) - quicksum(x[k,To,From] for To in Nodes[From].InLinks)
        if From == Commodities[k].From:
            Balance[k,From] = model.addConstr(balance, '=', Commodities[k].Quant,
                                              name = ''.join(['Balance(', str(k), ',', str(From), ',')']))
            = Demki
        elif From == Commodities[k].To:
            Balance[k,From] = model.addConstr(balance, '=', -Commodities[k].Quant,
                                              name = ''.join(['Balance(', str(k), ',', str(From), ',')']))
            = -Demki
        else:
            Balance[k,From] = model.addConstr(balance, '=', 0,
                                              name = ''.join(['Balance(', str(k), ',', str(From), ',')']))
            = 0
    # build 'capacity' constraints
    Capacity = {}
    for Arc in Arcs:
        Capacity[Arc.From,Arc.To] = model.addConstr(quicksum(x[k, Arc.From,Arc.To] for k in range(len(Commodities))), '<=', Arc.Capac,
                                                      name = ''.join(['Capacity(', str(Arc.From), ',', str(Arc.To), ')']))
    #save info to log file
    model.setParam("LogFile", 'log_file')
    #update gurobi with the constraints
    model.update()


$$\sum_{k \in K} x_m^k \leq Cap_m \quad \forall m \in L$$

```

MILP - Python & Gurobi - Constraints (2)

$$\sum_{m \in L_i^{\text{out}}} x_m^k - \sum_{m \in L_i^{\text{in}}} x_m^k$$

```
# build 'balance' constraints
# for .addConstr() check https://www.gurobi.com/documentation/current/refman/py\_model\_addconstr.html
Balance = {}
for k in range(len(Commodities)):
    for From in range(len(Nodes)):
        balance = quicksum(x[k,From>To] for To in Nodes[From].OutLinks) - quicksum(x[k,To,From] for To in Nodes[From].InLinks)
        if From == Commodities[k].From:
            Balance[k,From] = model.addConstr(balance, '=', Commodities[k].Quant,
                                              name = ''.join(['Balance(', str(k), ',', str(From), ',')']))
            = Demki
        elif From == Commodities[k].To:
            Balance[k,From] = model.addConstr(balance, '=', -Commodities[k].Quant,
                                              name = ''.join(['Balance(', str(k), ',', str(From), ',')']))
            = -Demki
        else:
            Balance[k,From] = model.addConstr(balance, '=', 0,
                                              name = ''.join(['Balance(', str(k), ',', str(From), ',')']))
            = 0

# build 'capacity' constraints
Capacity = {}
for Arc in Arcs:
    Capacity[Arc.From,Arc.To] = model.addConstr(quicksum(x[k, Arc.From,Arc.To] for k in range(len(Commodities))), '<=' , Arc.Capacity,
                                                name = ''.join(['Capacity(', str(Arc.From), ',', str(Arc.To), ')']))

# save info to log file
model.setParam("LogFile", 'log_file')
# update gurobi with the constraints
model.update()
```

$$\forall i \in N, k \in K$$

$$\sum_{k \in K} x_m^k \leq Cap_m$$

$$\forall m \in L$$

MILP - Results

Optimize a model with 27 rows, 28 columns and 84 nonzeros

Model fingerprint: 0x830f1a36

Coefficient statistics:

Matrix range [1e+00, 1e+00]

Objective range [1e+00, 8e+00]

Bounds range [0e+00, 0e+00]

RHS range [5e+00, 4e+01]

Presolve removed 20 rows and 17 columns

Presolve time: 0.00s

Presolved: 7 rows, 11 columns, 22 nonzeros

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	1.8993600e+02	5.016000e+00	0.000000e+00	0s
4	2.2000000e+02	0.000000e+00	0.000000e+00	0s

Solved in 4 iterations and 0.00 seconds (0.00 work units)

Optimal objective 2.200000000e+02

Arc(1 , 2)= 10

Arc(1 , 3)= 10

Arc(2 , 4)= 20

Arc(3 , 4)= 5

Arc(3 , 5)= 10

Arc(4 , 5)= 10

Objective Function = 220.0

Run Time = 0.002046823501586914

MILP - Results (2)

Optimize a model with 27 rows, 28 columns and 84 nonzeros

Model fingerprint: 0x830f1a36

Coefficient statistics:

Matrix range [1e+00, 1e+00]

Objective range [1e+00, 8e+00]

Bounds range [0e+00, 0e+00]

RHS range [5e+00, 4e+01]

Presolve removed 20 rows and 17 columns

Presolve time: 0.00s

Presolved: 7 rows, 11 columns, 22 nonzeros

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	1.8993600e+02	5.016000e+00	0.000000e+00	0s
4	2.2000000e+02	0.000000e+00	0.000000e+00	0s

Solved in 4 iterations and 0.00 seconds (0.00 work units)

Optimal objective 2.200000000e+02

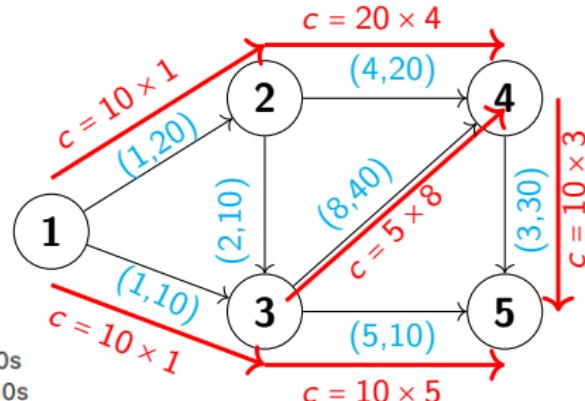
Arc(1, 2) = 10
Arc(1, 3) = 10
Arc(2, 4) = 20
Arc(3, 4) = 5
Arc(3, 5) = 10
Arc(4, 5) = 10

Objective Function = 220.0

Run Time = 0.002046823501586914

→ Optimal solution

Objective
Function
Value



Commodities		
From (<i>i</i>)	To (<i>j</i>)	Quantity (<i>k</i>)
1	4	15
1	5	5
2	5	10
3	5	5

MILP - Output

A log file is produced in the folder you are running the model script.

Want to know what the log file tells you? ([Link](#))

```
MILP Solve
Academic license - for non-commercial use only
Parameter OutputFlag unchanged
    Value: 1 Min: 0 Max: 1 Default: 1
Changed value of parameter TimeLimit to 60.0
    Prev: 1e+100 Min: 0.0 Max: 1e+100 Default: 1e+100
Optimize a model with 79 rows, 520 columns and 1825 nonzeros
Model has 432 general constraints
Variable types: 0 continuous, 520 integer (507 binary)
Coefficient statistics:
    Matrix range      [1e+00, 5e+02]
    Objective range   [1e+03, 6e+03]
    Bounds range      [1e+00, 4e+00]
    RHS range         [1e+00, 8e+02]

Loaded MIP start with objective 25408.6

Presolve added 803 rows and 0 columns
Presolve removed 0 rows and 40 columns
Presolve time: 0.02s
Presolved: 882 rows, 480 columns, 4212 nonzeros
Variable types: 0 continuous, 480 integer (468 binary)

Root relaxation: objective 1.789331e+04, 80 iterations, 0.00 seconds

          Nodes    |   Current Node   |   Objective Bounds   |   Work
Expl Unexpl |   Obj  Depth IntInf   Incumbent   BestBd   Gap |   It/Node Time
          0     0 17893.3147   0   24 25408.5806 17893.3147  29.6%  -    0s
          0     0 17893.3147   0   24 25408.5806 17893.3147  29.6%  -    0s
          0     17993.4331   0   24 25408.5806 17993.4331  29.2%  -    0s
          0     18093.5516   0   24 25408.5806 18093.5516  28.8%  -    0s
          0     18293.7884   0   28 25408.5806 18293.7884  28.0%  -    0s
          0     18293.7884   0   26 25408.5806 18293.7884  28.0%  -    0s
          0     18349.2866   0   34 25408.5806 18349.2866  27.8%  -    0s
          0     18349.2866   0   31 25408.5806 18349.2866  27.8%  -    0s
          0     18349.2866   0   34 25408.5806 18349.2866  27.8%  -    0s
          0     18349.2866   0   31 25408.5806 18349.2866  27.8%  -    0s
          0     18349.2866   0   38 25408.5806 18349.2866  27.8%  -    0s
          0     18349.2866   0   38 25408.5806 18349.2866  27.8%  -    0s
          0     2 18349.2866   0   24 25408.5806 18349.2866  27.8%  -    0s
H 131    113                           24594.455369 18608.4827  24.3%  12.8  0s
  5583   1228 22313.7026   24   27 24594.4554 21483.7565 12.6%  11.1  5s

Cutting planes:
    Gomory: 1
    Implied bound: 4
    MIR: 18

Explored 13704 nodes (140151 simplex iterations) in 9.04 seconds
Thread count was 4 (of 4 available processors)

Solution count 2: 24594.5 25408.6

Optimal solution found (tolerance 1.00e-04)
Best objective 2.459445536866e+04, best bound 2.459445536866e+04, gap 0.0000%
```

MILP - Tableau

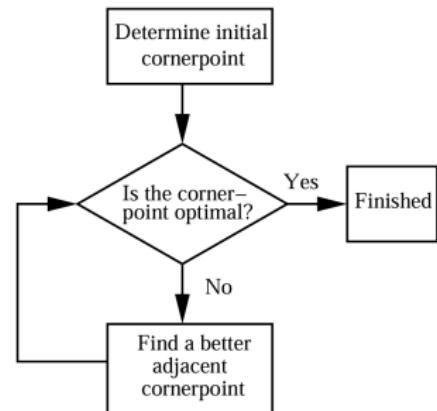
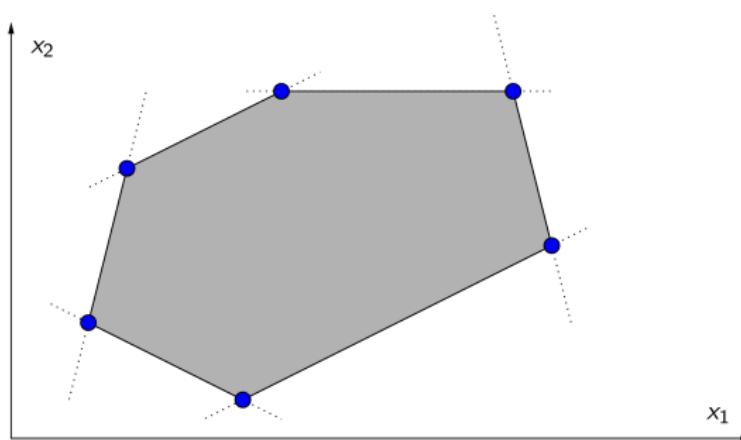
GUROBI is reading our input and defining our problem following a tableau (SIMPLEX method): [Link](#)

	k=1					k=2					k=3					k=4					RHS		
	1-2	1-3	2-3	2-4	3-4	3-5	4-5	1-2	1-3	2-3	2-4	3-4	3-5	4-5	1-2	1-3	2-3	2-4	3-4	3-5	4-5		
1	1	1																				= d ₁	
2	-1		1	1																		= 0	
3		-1	-1		1	1																= 0	
4			-1	-1			1															= - d ₁	
5				-1	-1																	= 0	
1								1	1													= d ₂	
2								-1	1	1												= 0	
3								-1	-1		1	1										= 0	
4									-1	-1			1									= 0	
5										-1	-1											= - d ₂	
1												1	1									= 0	
2												-1	1	1								= d ₃	
3												-1	-1	1	1							= 0	
4													-1	-1	1	1						= 0	
5														-1	-1							= - d ₃	
1																	1	1				= 0	
2																	-1	1	1			= 0	
3																	-1	-1	1	1		= d ₄	
4																		-1	-1	1			= 0
5																			-1	-1			= d ₄
1-2	1							1	1								1	1				$\leq u_{12}$	
1-3		1							1									1					$\leq u_{13}$
2-3			1							1									1				$\leq u_{23}$
2-4				1							1									1			$\leq u_{24}$
3-4					1							1									1		$\leq u_{34}$
3-5						1							1									1	$\leq u_{35}$
4-5							1																$\leq u_{45}$
Cost Variable	C_{12}^1	C_{13}^1	C_{23}^1	C_{24}^1	C_{34}^1	C_{35}^1	C_{45}^1	C_{12}^2	C_{13}^2	C_{23}^2	C_{23}^2	C_{34}^2	C_{35}^2	C_{45}^2	C_{12}^3	C_{23}^3	C_{23}^3	C_{24}^3	C_{34}^3	C_{35}^3	C_{45}^3	C_{12}^4	
	X_{12}^{-1}	X_{13}^{-1}	X_{23}^{-1}	X_{24}^{-1}	X_{34}^{-1}	X_{35}^{-1}	X_{45}^{-1}	X_{12}^{-2}	X_{13}^{-2}	X_{23}^{-2}	X_{24}^{-2}	X_{34}^{-2}	X_{35}^{-2}	X_{45}^{-2}	X_{12}^{-3}	X_{23}^{-3}	X_{23}^{-3}	X_{24}^{-3}	X_{34}^{-3}	X_{35}^{-3}	X_{45}^{-3}	X_{12}^{-4}	

MILP - Simplex Method

The idea of the Simplex method is to iteratively search along edges to other corner points for a better objective function value

If an optimal solution exist, one optimal solution will be an extreme point

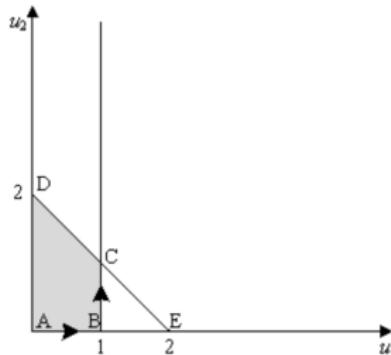
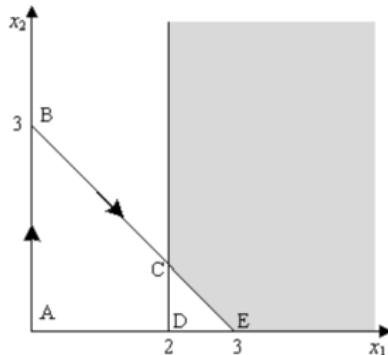


MILP - Simplex Method & Dual Theory

For maximization or minimization problems, the simplex method may be tedious.

Every LP problem (known as the primal) has its dual. If the primal is maximization, then the dual is minimization, and vice-versa. The solutions for the dual problem are the same as for the primal.

The simplex method is applied simultaneously to its dual problem. The two methods move in complete sequence, obtaining feasible solutions to the dual problems with each iteration.



Banciu, Mihai. (2011). Dual Simplex.

Outline - Lecture 2

- ① Multi-commodity Flow Models
 - a Decision Variables & Parameters
 - b Objective Function
 - c Constraints
 - d Problem Formulation
 - e Example
- ② Mixed Integer Linear Programming
 - a Python & Gurobi
 - b Results
 - c Output
 - d Tableau
 - e Simplex
 - f Simplex & Dual Theory
- ③ Homework

Homework

Other possible constraints' formulations might be needed for other problems. How would you model these constraints:

$$\sum_i Ax_{ij} \geq B \quad \forall j \in \mathbf{N}$$

$$\sum_i Ax_{ij} \leq \sum_i y_i \quad \forall j \in \mathbf{N}$$

$$\sum_i \sum_j B_{ij} x_{ij} \leq Ay_i$$

Take inspiration from *Tutorial: Getting Started with the Gurobi Python API*

Airline Planning and Optimisation - AE4423-20

Lecture 3 Network and Fleet Planning

Marta Ribeiro
Assistant Professor
Air Transport & Operation
Control & Operations

Delft University of Technology, The Netherlands

November, 2024

Course Information - Content

Management

Planning structure
Performance Indicators
Demand and Supply
Market share

Tactical Planning

Passenger Mix Flow
Fleet Assignment
Aircraft Rotation
Crew Scheduling
Maintenance



Strategical Planing

Network Structure
Demand Forecast
Network Planning
Fleet Planning

Operations Research

MILP Models
Multi-commodity Flow Problems
Shortest-path Algorithms
Column Generation Algorithm
Dynamic Programming

Outline - Lecture 3

① Network Development Decisions

- ② a Route planning factors
- ② b Point to point (P2P) vs Hub-&-Spoke (H&S)
- ② c Hub-&-Spoke (H&S)
- ② d Point to point (P2P)

② Fleet Planning Decisions

- ③ a Problem Characteristics
- ③ b Aircraft Characteristics
- ③ c Economic Factors
- ③ d Leasing Schemes

③ Network and Fleet Modelling

- ④ a Point to Point Network model
- ④ b Hub-&-Spoke Network Model
- ④ c Fleet & Network Model

Network Development - Route Planning Factors

'Given a fleet plan, to select the routes to be flown'

But, sometimes ...

'to identify the fleet needed to operate a network'

Ideally, these two long-term (strategic) decision should be integrated

Economical factors:

- Passengers & cargo demand (direct and connecting)
- Competition
- Need to invest in new facilities and personnel

Technical factors:

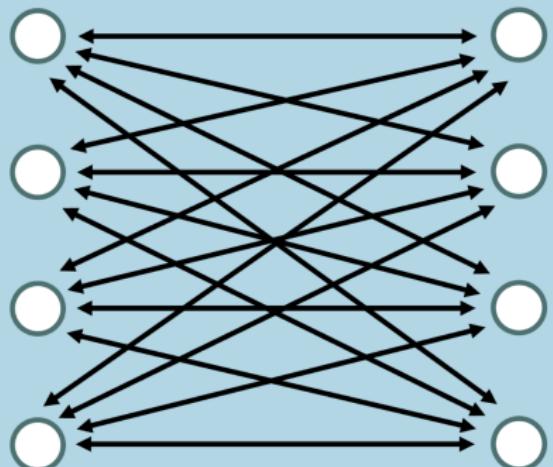
- Aircraft characteristics (e.g., range, capacity, fuel efficiency)
- Airport requirements (e.g., runway or gate constraints)

Others factors:

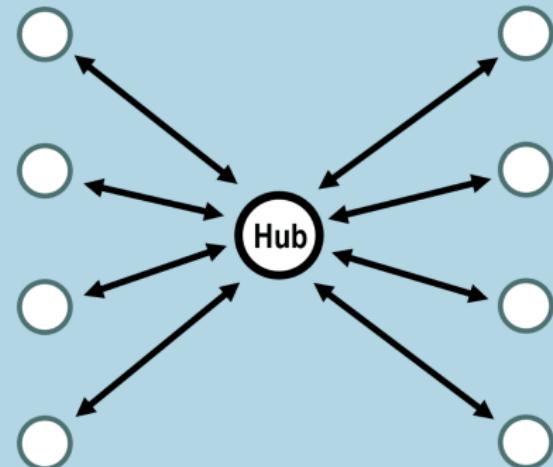
- Airport slots
- Regulations and Air Service Agreements (ASAs)

Network Development - Point to Point vs Hub-&-Spoke

POINT-TO-POINT



HUB-AND-SPOKE



Source: transportgeography.org

Network Development - Point to Point vs Hub-&-Spoke (2)



Network Development - Point to Point vs Hub-&-Spoke (2)

Delta



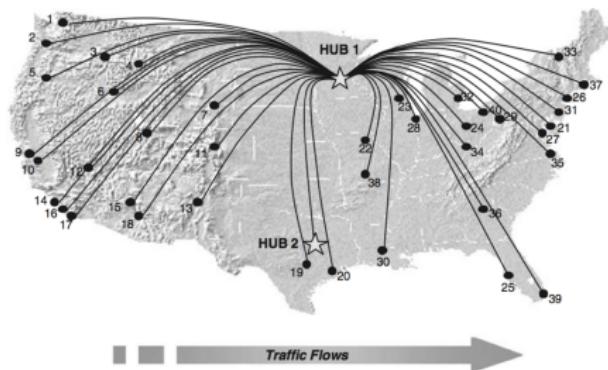
Ryanair



Note: the difference is more related to the service and not just topology!

Network Development - Hub-&-Spoke

Airline with a hub (HUB 1) and serving 40 other airports:



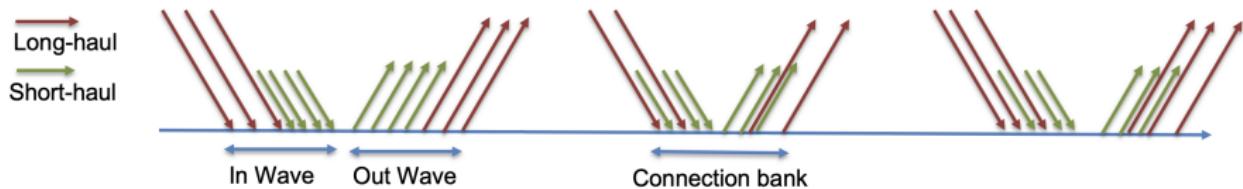
- 20 flights arriving followed by 20 departing flights
- Each flight leg arriving or departing the hub simultaneously serves 21 O-D markets
- 40 flights \Rightarrow 20 aircraft \Rightarrow 440 O-D Markets
- A P2P network would involve 440 flight legs and hundreds of aircraft

Source: Belobaba et al, The Global Airline Industry, 2009

Network Development - Hub-&-Spoke (2)

Connecting banks (H&S networks):

- 'Waves' of inbound and outbound flights
- They last from approximately **1.5 to 2 hours**



Schiphol/KLM:

- 4 main connecting banks per day: 7:00; 9:30; 14:00; 20:00
- Plus 3 smaller connecting banks: 11:30; 16:30; 21:30

Network Development - Point to Point (P2P)

Low-cost carriers operate under **P2P** networks:

- Explore the revenue of non-stop flights
- Do not sell multi-flight tickets
- Aim for an higher aircraft utilisation
- Follow two strategies to 'densify' market demand:
 - High frequency by providing several flights per day
 - Or, most commonly, decrease trip fares
- Reduce airport costs by operating to secondary airports or by reducing the service needs at the airports

Outline - Lecture 3

① Network Development Decisions

- ② a Route planning factors
- ② b Point to point (P2P) vs Hub-&-Spoke (H&S)
- ② c Hub-&-Spoke (H&S)
- ② d Point to point (P2P)

② Fleet Planning Decisions

- ③ a Problem Characteristics
- ③ b Aircraft Characteristics
- ③ c Economic Factors
- ③ d Leasing Schemes

③ Network and Fleet Modelling

- ④ a Point to Point Network Model
- ④ b Hub-&-Spoke Network Model
- ④ c Fleet & Network Model

Fleet Planning - Problem Characteristics

The fleet planning process is highly influenced by **cash availability** and **market opportunities**

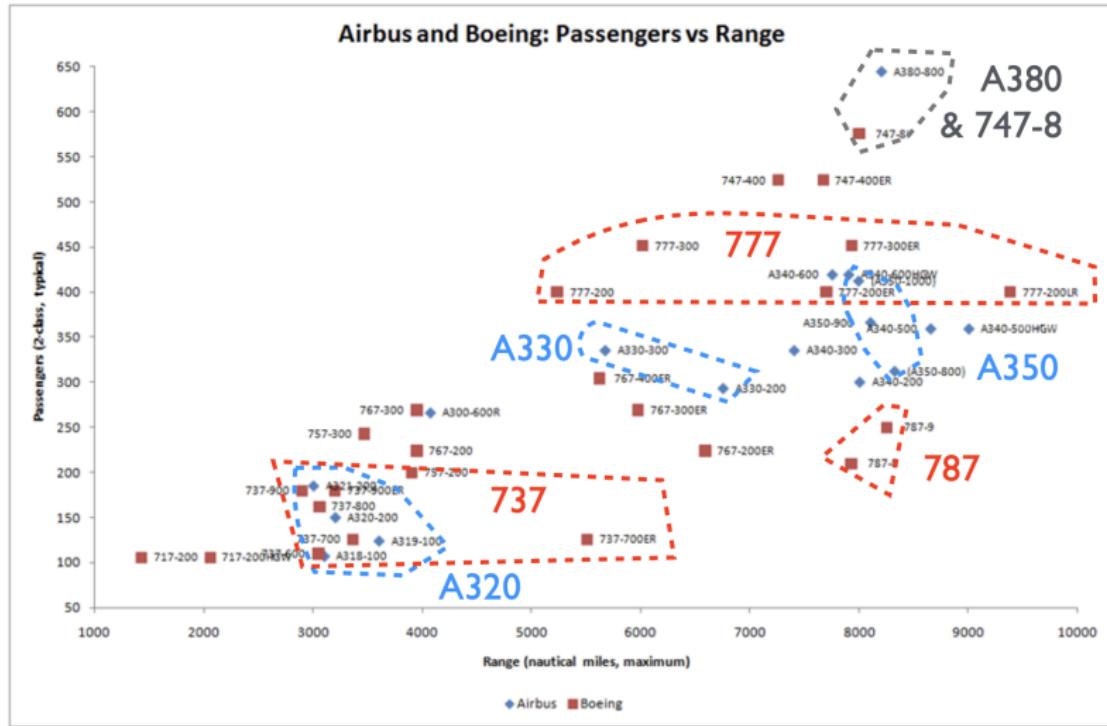
Long-term Decision Process:

- Manufacturers:
 - 5 - ? years aircraft definition
 - 5 years aircraft development
 - 20 - 30 years production
- Airlines:
 - Large capital investment
 - Long-term financial impact
 - Long-term operations

Key Decision Criteria:

- Performance:
 - Fuel consumption
 - Engines
 - Turn-around times
- Range
- Capacity
- Comfort
- Commonality (fleet)
- Environmental impact
- Market and infrastructure

Fleet Planning - Aircraft Characteristics - Range vs Seats

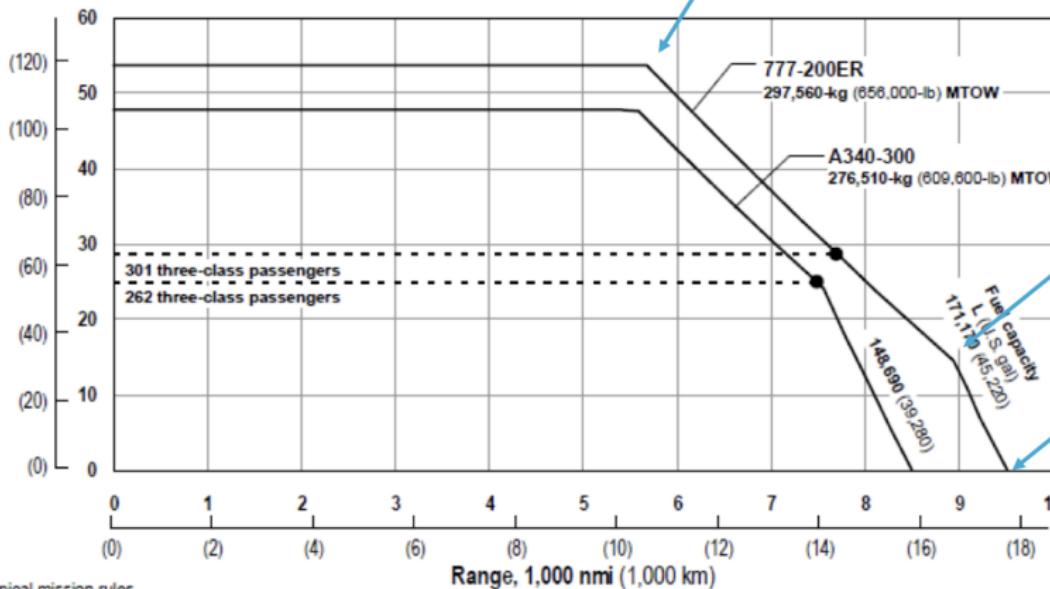


Source: wikipedia

Fleet Planning - Aircraft Characteristics - Payload Range

777-200ER versus A340-300

Payload, 1,000 kg (1,000 lb)



- Typical mission rules.
- Three-class seating.
- 200-nmi alternate.
- 95.25 kg (210 lb) per passenger.

Source: airliners.net

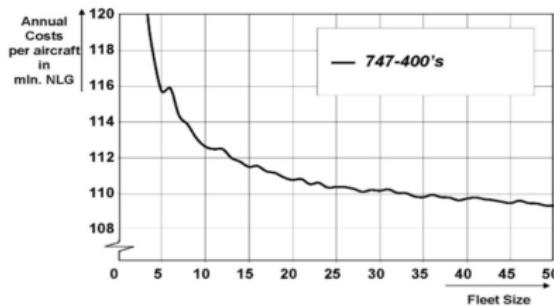
Fleet Planning - Economic factors

Financing from internal or external sources:

- Cash deposits, loans, or equity
- Lease vs. ownership

Determine **cost** and **revenues**:

- Up-front: purchase price, spare parts (engine and aircraft), ground equipment and training
- New vs. second-hand aircraft
- Fleet commonality to reduce maintenance and operating costs
 - 27 days of retraining pilots from Airbus A320 to A330
 - Economies of scale in maintenance



Source: TUD MSc thesis work - Annual landing gear replacement costs as a function of fleet size

Fleet Planning - Leasing Schemes

Aircraft leasing is a contract between a **lessor** and a **lessee**. The lessee:

- Does not own the aircraft
- Selects the aircraft specification
- Makes specific payments to the lessor for a given period
- Has an exclusive use of the aircraft for that same period
- Has operational control over the flights

Leasing (current trend) vs purchasing:

- Leasing gives more flexibility
- Leasing does not involve intensive capital investment, but does not provide profit for final sale either
- Over a long period of time (≈ 10 years), leasing may become more expensive than purchasing

Fleet Planning - Leasing Schemes (2)

Wet Lease (or ACMI - Aircraft, Crew, Maintenance & Insurance)	Usually between airlines in busy periods The lessor provides the AC, crew (cabin, cockpit and engineers), all maintenance, and insurance Generally lasts for 1-24 months
Dry Lease	The lease of the AC (without "CMI") Operating lease: <ul style="list-style-type: none">• Usually ranges from 2 to 7 years• AC does not appear on the lessee's balance sheet Finance lease (or capital lease): <ul style="list-style-type: none">• From 5 to 20 years• AC appears on the lessee's balance sheet (viewed as a purchase)
Damp Lease	Similar to wet lease but without cabin crew

Outline - Lecture 3

① Network Development Decisions

- ② a Route planning factors
- ② b Point to point (P2P) vs Hub-&-Spoke (H&S)
- ② c Hub-&-Spoke (H&S)
- ② d Point to point (P2P)

② Fleet Planning Decisions

- ③ a Problem Characteristics
- ③ b Aircraft Characteristics
- ③ c Economic Factors
- ③ d Leasing Schemes

③ Network and Fleet Modelling

- ④ a Point to Point Network Model
- ④ b Hub-&-Spoke Network Model
- ④ c Fleet & Network Model

Network and Fleet Modelling

Contents:

- | | |
|-----------------------------------|------------------------|
| ① Model 1: Point-to-Point Network | Simplest Model |
| ② Model 2: Hub-&-Spoke Network | Add Connections |
| ③ Model 3: Fleet & Network | Add Fleet |
| ④ Simplifications | If Needed |

Network and Fleet Modelling

Contents:

- | | |
|--|--|
| <ul style="list-style-type: none">① Model 1: Point-to-Point Network
② Model 2: Hub-&-Spoke Network
③ Model 3: Fleet & Network
④ Simplifications | <p>Simplest Model</p> <p>Add Connections</p> <p>Add Fleet</p> <p>If Needed</p> |
|--|--|

Modelling - Model 1: Point-to-Point

The problem:

- Given a set of airports from which an airline can operate
 - Knowing the demand in each specific flight-leg
 - Having a set of aircraft (homogenous fleet), with specific characteristics (e.g., speed, seats, block hours)
-
- Define which flight legs to operate
 - Estimated number of passengers to transport
 - Define the flight frequency per leg

Modelling - Model 1: Point-to-Point (2)

Objective Function:

- Minimize costs (satisfying the demand)
- Maximize revenues
- Maximize profit

(OF)

Subject to:

- Demand verification: $\#pax \leq \text{demand}$ (C1)
- Capacity: $\#pax \text{ in each leg} \leq \text{seats available per leg}$ (C2)
- Continuity constraint: $\#\text{AC inbound} = \#\text{AC}$
outbound (per airport, per AC type) (C3)
- AC Productivity: $\text{hours of operation} \leq \text{Block Time}$
(BT) $\times \# \text{ AC}$ (C4)

Modelling - Model 1: Point-to-Point (3)

Sets: N : set of airports

Decision variables:

x_{ij} : flow from airport i to airport j

z_{ij} : number of flights from airport i to airport j

Parameters:

q_{ij} : travel demand between airport i to airport j

d_{ij} : distance between airports i and j

Yield : revenue per Revenue Passenger Kilometers (RPK) flown (average yield)

s : number of seats per aircraft

CASK : unit operation cost per per available seat per kilometre (ASK) flown

sp : speed of the aircraft

LF : average load factor

AC : number of aircraft

LTO : landing and take-off time

BT : aircraft avg. utilisation time

Modelling - Model 1: Point-to-Point (4)

Max Profit:

$$\sum_{i \in N} \sum_{j \in N} \left(\overbrace{Yield \times d_{ij} \times x_{ij}}^{\text{Revenue}} - \underbrace{CASK \times d_{ij} \times s \times z_{ij}}_{\text{Cost of Operation}} \right) \quad (\text{OF})$$

Subject to:

$$x_{ij} \leq q_{ij}, \quad \forall i, j \in N \quad (\text{C1})$$

$$x_{ij} \leq z_{ij} \times s \times LF, \quad \forall i, j \in N \quad (\text{C2})$$

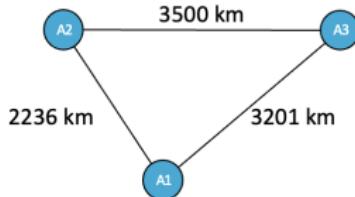
$$\sum_{j \in N} z_{ij} = \sum_{j \in N} z_{ji}, \quad \forall i \in N \quad (\text{C3})$$

$$\sum_{i \in N} \sum_{j \in N} \left(\frac{d_{ij}}{sp} + LTO \right) \times z_{ij} \leq BT \times AC \quad (\text{C4})$$

Modelling - Model 1: Point-to-Point - Problem

An airline will start operating from the 3 airports. The demand per flight leg and aircraft characteristics are provided below. Determine:

- ① The frequency in each flight leg assuming that you can operate 2 aircraft with the characteristics presented in the following table and you expect a revenue of 0.18 €/RPK
- ② Idem, but only with 1 aircraft
- ③ Idem (2 AC), but a reduction in revenue to 0.15 €/RPK

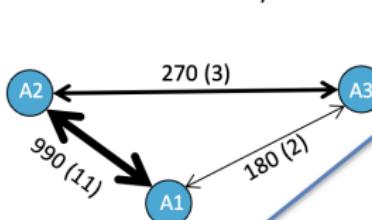


Airport	Demand (pax/week)		
	A1	A2	A3
A1	0	1000	200
A2	1000	0	300
A3	200	300	0

Aircraft		
Parameters		
CASK	0,12	[\$/ASK]
LF	0,75	[%]
Seats	120	[units]
Speed	870	[km/h]
LTO	20	[min]
BT	10	[h/day]
Fleet	2	[units]

Modelling - Model 1: Point-to-Point - Solution

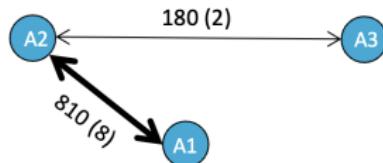
(a) OF: 149 400 € /week
AC utiliz.: 75,8 %



(c) OF: NA (not profitable to fly)



(b) OF: 97 649 € /week
AC utiliz.: 99,6 %



Legend:
Pax (flights)

Network and Fleet Modelling

Contents:

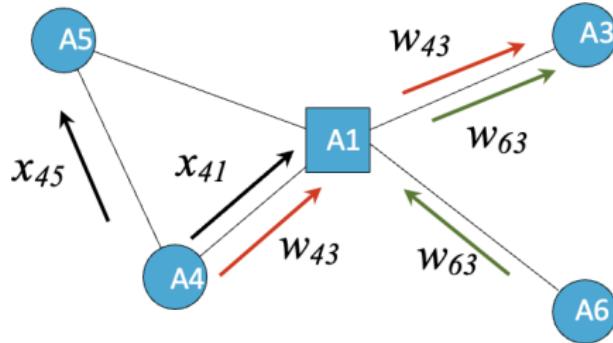
- | | |
|-----------------------------------|------------------------|
| ① Model 1: Point-to-Point Network | Simplest Model |
| ② Model 2: Hub-&-Spoke Network | Add Connections |
| ③ Model 3: Fleet & Network | Add Fleet |
| ④ Simplifications | If Needed |

Modelling - Model 2: Hub-&-Spoke Network

Several airlines provide the possibility for passengers to connect at a hub airport. In Hub-&-Spoke networks:

- Demand is forecast according to the OD market
- The flight leg demand depends on our decisions
- Hub airports guarantee connections between flights

Let's address the dichotomy between demand and supply:



Modelling - Model 2: Hub-&-Spoke Network (2)

Sets:

N : set of airports

Decision variables:

x_{ij} : direct flow from airport i to airport j

z_{ij} : number of flights from airport i to airport j

w_{ij} : flow from airport i to airport j that transfers at the hub

Parameters:

q_{ij} : travel demand between airport i to airport j

d_{ij} : distance between airports i and j

Yield : revenue per Revenue Passenger Kilometers (RPK) flown (average yield)

s : number of seats per aircraft

CASK : unit operation cost per available seat per kilometre (ASK) flown

sp : speed of the aircraft

LF : average load factor

AC : number of aircraft

LTO : landing and take-off time

BT : aircraft avg. utilisation time

g_k : 0 if a hub is located at airport k ; 1 otherwise

Modelling - Model 2: Hub-&-Spoke Network (3)

Max Profit:

$$\sum_{i \in N} \sum_{j \in N} [Yield \times d_{ij} \times (x_{ij} + w_{ij}) - CASK \times d_{ij} \times s \times z_{ij}] \quad (\text{OF})$$

Subject to:

$$x_{ij} + w_{ij} \leq q_{ij}, \quad \forall i, j \in N$$

$$w_{ij} \leq q_{ij} \times g_i \times g_j, \quad \forall i, j \in N$$

$$x_{ij} + \sum_{m \in N} w_{im} \times (1 - g_j) + \sum_{m \in N} w_{mj} \times (1 - g_i) \leq z_{ij} \times s \times LF, \quad \forall i, j \in N$$

$$\sum_{j \in N} z_{ij} = \sum_{j \in N} z_{ji}, \quad \forall i \in N$$

$$\sum_{i \in N} \sum_{j \in N} \left(\frac{d_{ij}}{sp} + LTO \right) \times z_{ij} \leq BT \times AC$$

(C1) All flow from each airport leave the airport, either through a hub or not

(C1*) Only consider 'transfer' passengers if the hub is not origin or destination. Otherwise, it is considered direct flow

(C2) Capacity verification in each flight leg

(C3) Balance between incoming and outgoing flights at each node

(C4) Use of aircraft limited to the number of aircraft and the block hours associated

Modelling - Model 2: Hub-&-Spoke Network (4)

How do constraints C2 work?

$$x_{ij} + \sum_{m \in N} w_{im}(1 - g_j) + \sum_{m \in N} w_{mj}(1 - g_i) \leq z_{ij} \times x \times LF, \forall i, j \in N$$

- For $i = 4$ and $j = 1$:

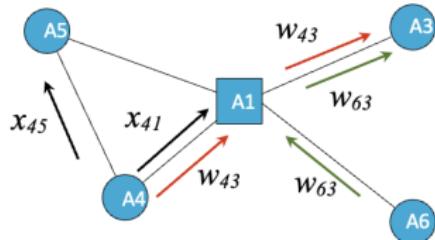
$$\cancel{x_{4,1}} + [(w_{4,1} + w_{4,2} + w_{4,3} + \dots) \times (1 - 0)] + [(w_{1,1} + w_{2,1} + w_{3,1} + \dots) \times (1 - 1)] \leq z_{4,1} \times x \times LF$$

- For $i = 1$ and $j = 3$:

$$\cancel{x_{1,3}} + [(w_{1,1} + w_{1,2} + w_{1,3} + \dots) \times (1 - 1)] + [(w_{1,3} + w_{2,3} + w_{3,3} + \dots) \times (1 - 0)] \leq z_{1,3} \times x \times LF$$

- For $i = 4$ and $j = 5$:

$$\cancel{x_{4,5}} + [(w_{4,1} + w_{4,2} + w_{4,3} + \dots) \times (1 - 1)] + [(w_{1,5} + w_{2,5} + w_{3,5} + \dots) \times (1 - 1)] \leq z_{4,5} \times x \times LF$$



Modelling - Model 2: Hub-&-spoke Network - Problem

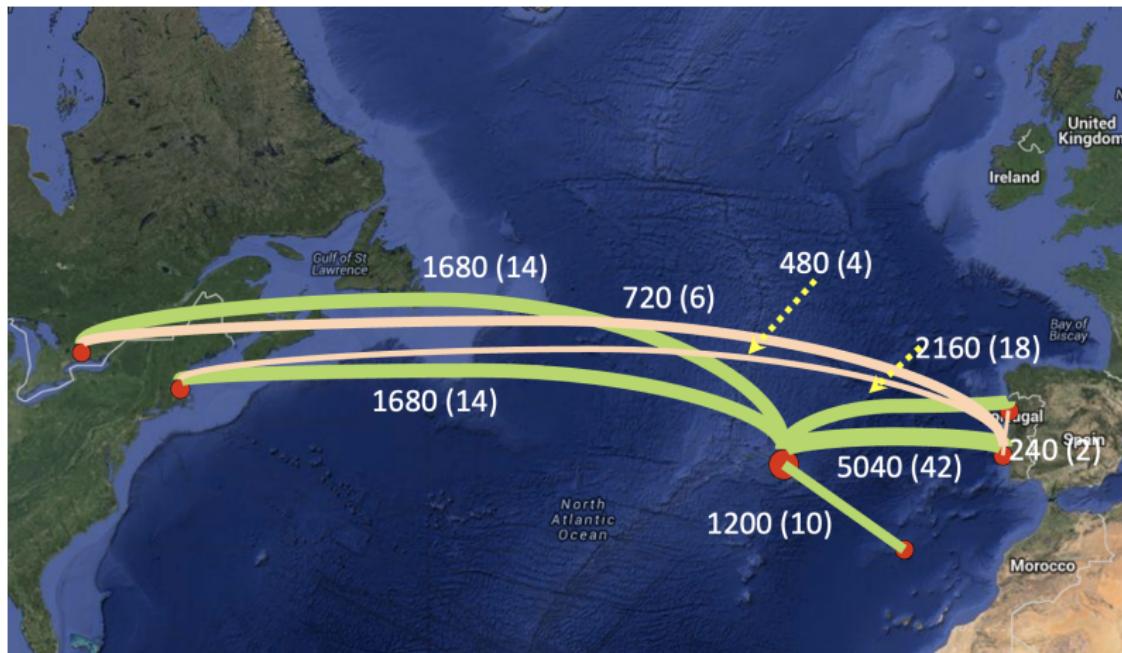
The FlyAtlantic Airline is starting its operations with a hub in the Azores Islands. The company wants to connect the Portuguese main airports (Lisbon, Oporto, Funchal, Ponta Delgada) with Boston and Toronto. The airline needs to define the most profitable network and the frequency of flights. The airline has 4 aircraft of the same type. The company forecasts an average revenue 0.16 €/RPK. Propose a network based on the forecasted average weekly demand provided.

Airport	PDL	Distances (km)				
		LIS	OPO	FNC	YTO	BOS
PDL	0	1461	1536	975	4545	3888
LIS	1461	0	336	973	5790	5177
OPO	1536	336	0	1244	5671	5081
FNC	975	973	1244	0	5515	4851
YTO	4545	5790	5671	5515	0	691
BOS	3888	5177	5081	4851	691	0

Parameters		
CASK	0,12	[\$/ASK]
LF	0,80	[%]
Seats	150	[units]
Speed	890	[km/h]
LTO	20	[min]
BT	13	[h/day]
Fleet	4	[units]

Airport	Demand (pax/week)					
	PDL	LIS	OPO	FNC	YTO	BOS
PDL	0	2509	1080	558	770	713
LIS	2509	0	216	112	360	333
OPO	1080	216	0	78	46	43
FNC	558	112	78	0	32	30
YTO	770	360	46	32	0	70
BOS	713	333	43	30	70	0

Modelling - Model 2: Hub-&-spoke Network - Solution



Legend: Pax (flights)

OF: 316 821 €/week

AC Utilisation: 94.33%

Network and Fleet Modelling

Contents:

- | | |
|-----------------------------------|------------------------|
| ① Model 1: Point-to-Point Network | Simplest Model |
| ② Model 2: Hub-&-Spoke Network | Add Connections |
| ③ Model 3: Fleet & Network | Add Fleet |
| ④ Simplifications | If Needed |

Modelling - Model 3: Fleet & Network

The problem:

- Given a set of airports from which an airline can operate
 - Knowing the demand in each specific flight-leg
 - **Having a set of different aircraft types available**, with specific characteristics (e.g., speed, seats, block hours)
-
- Define which flight legs to operate
 - Estimated number of passengers to transport
 - Define the flight frequency per leg
 - **Define the fleet size and composition**

Modelling - Model 3: Fleet & Network (2)

Objective Function:

- Minimize costs (satisfying the demand)
 - Maximize revenues
 - Maximize profit
- (OF)

Subject to:

- Demand verification: $\#pax \leq \text{demand}$ (C1)
- Capacity: $\#pax \text{ in each leg} \leq \text{seats available per leg}$ (C2)
- Continuity constraint: $\#AC \text{ inbound} = \#AC \text{ outbound}$ (per airport, per AC type) (C3)
- AC Productivity: $\text{hours of operation} \leq BT \times \# AC$ (C4)
- Range Constraint: $\text{Range} \leq \text{distance} \Rightarrow \text{no flights}$ (C5)
- Budget Constraint: $\#AC \times \text{Cost} \leq \text{Budget}$ (C6)

Modelling - Model 3: Fleet & Network (3)

Sets:

N : set of airports

K : set of aircraft types

Decision variables:

x_{ij} : direct flow from airport i to airport j

z_{ij}^k : number of flights from airport i to airport j for aircraft type k

w_{ij} : flow from airport i to airport j that transfers at the hub

AC^k : number of aircraft type k

Parameters:

q_{ij} : travel demand between airport i to airport j

d_{ij} : distance between airports i and j

$Yield$: revenue per Revenue Passenger Kilometers (RPK) flown (average yield)

s^k : number of seats per aircraft type k

$CASK^k$: unit operation cost per available seat per kilometre (ASK) flown by aircraft type k

sp^k : speed of the aircraft type k

LF : average load factor

AC : number of aircraft

LTO : landing and take-off time

BT^k : aircraft type k avg. utilisation time

g_k : 0 if a hub is located at airport k ; 1 otherwise

C^k : cost of purchasing aircraft type k

Modelling - Model 3: Fleet & Network (4)

Max Profit:

$$\sum_{i \in N} \sum_{j \in N} [Yield \times d_{ij} \times (x_{ij} + w_{ij}) - \sum_{k \in K} (CASK^k \times d_{ij} \times s^k \times z_{ij}^k)] \quad (\text{OF})$$

Subject to:

$$x_{ij} + w_{ij} \leq q_{ij}, \quad \forall i, j \in N \quad (\text{C1})$$

$$w_{ij} \leq q_{ij} \times g_i \times g_j, \quad \forall i, j \in N \quad (\text{C1}^*)$$

$$x_{ij} + \sum_{m \in N} w_{im} \times (1 - g_j) + \sum_{m \in N} w_{mj} \times (1 - g_i) \leq \sum_{k \in K} z_{ij}^k \times s^k \times LF, \quad \forall i, j \in N \quad (\text{C2})$$

$$\sum_{j \in N} z_{ij}^k = \sum_{j \in N} z_{ji}^k, \quad \forall i \in N, k \in K \quad (\text{C3})$$

$$\sum_{i \in N} \sum_{j \in N} \left(\frac{d_{ij}}{sp^k} + LTO \right) \times z_{ij}^k \leq BT^k \times AC^k, \quad \forall k \in K \quad (\text{C4})$$

$$z_{ij}^k \leq a_{ij}^k, \quad \forall i, j \in N, k \in K \quad \rightarrow \quad a_{ij}^k = \begin{cases} 10000 & \text{if } d_{ij} \leq R^k \\ 0 & \text{otherwise} \end{cases}$$

$$\sum_{k \in K} C^k \times AC^k \leq B$$

(C5) Aircraft range used to define matrix a_{ij}^k and constrain frequency to range limits

(C6) Investment costs cannot be higher than the budget available

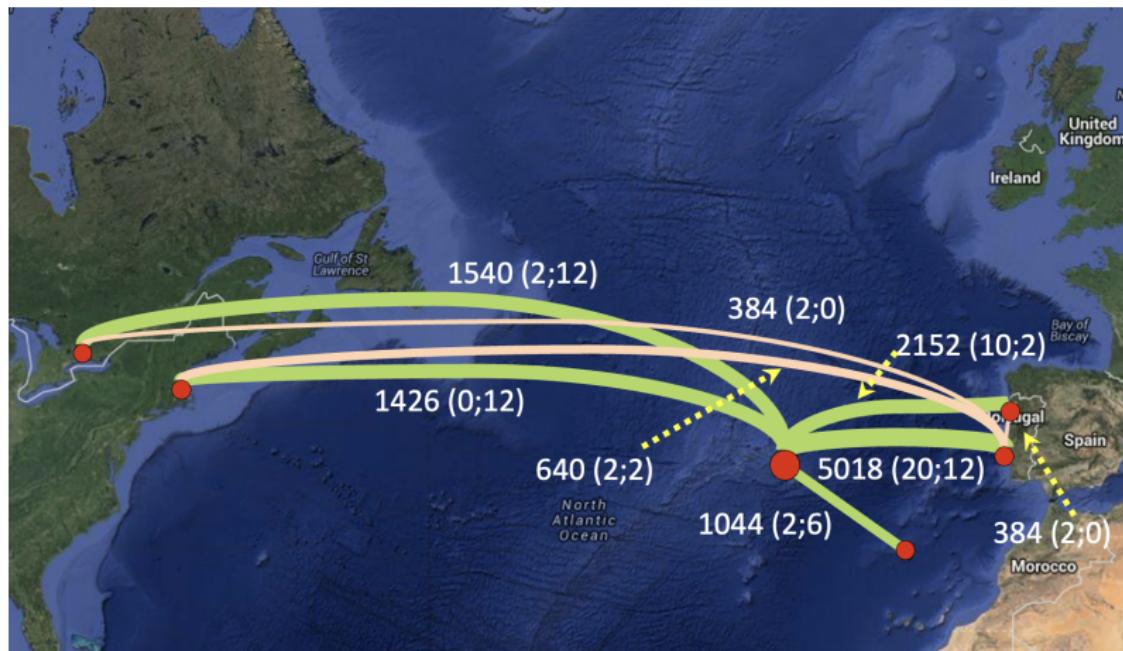
Modelling - Model 3: Fleet & Network - Problem

What if the FlyAtlantic Airline would have a fleet of different aircraft. How should we model a multi-fleet network model (aka, multi-class network model)?

Would the results differ from the results obtained in Problem 2?

Parameters	A310	A320	Units
CASK	0.12	0.11	[\$/ASK]
LF	0,8	0.8	[%]
Seats	240	160	
Range	9600	5400	[km]
Speed	900	870	[km/h]
LTO	20	12	[min]
BT	14	12	[h/day]
Fleet	2	2	

Modelling - Model 3: Fleet & network - Solution



Legend: Pax (flights A310; A320)

OF: 533 985 €/week

AC Utilisation: A310 - 51.55% ; A320 - 99.9%

Network and Fleet Modelling

Contents:

- | | |
|---|--|
| <ul style="list-style-type: none">① Model 1: Point-to-Point Network② Model 2: Hub-&-Spoke Network③ Model 3: Fleet & Network④ Simplifications | <p>Simplest Model</p> <p>Add Connections</p> <p>Add Fleet</p> <p>If Needed</p> |
|---|--|

Modelling - Simplifications

The previous models represent an airline network development decision process in a simplified way. For instance:

- **Demand is static** - demand does not vary with the solutions obtained (e.g., frequency, direct vs connecting flight)
- **No competition** - market share is assumed to be already considered when defining demand values
- **No passengers choice** - demand is reacting to the decisions according to the objective of the airline (i.e., maximise profit)
- **Possible aircraft routing discontinuity** - a total 'time-budget' is considered for the entire fleet without checking routing feasibility
- **Single scenario for a static future** - the uncertainty of the future network is not considered and the model runs for a single time period in the future.

Airline Planning and Optimisation - AE4423-20

Lecture 4 Passenger Mix Flow

Marta Ribeiro
Assistant Professor
Air Transport & Operation
Control & Operations

Delft University of Technology, The Netherlands

November, 2024

Course Information - Content

Management

Planning Framework
Performance Indicators
Demand and Supply
Market Share

Tactical Planning

Passenger Mix Flow
Fleet Assignment
Aircraft Rotation
Crew Scheduling
Maintenance



Strategical Planing

Network Structure
Demand Forecast
Network Planning
Fleet Planning

Operations Research

MILP Models
Multi-commodity Flow Problems
Shortest-path Algorithms
Column Generation Algorithm
Dynamic Programming

Outline - Lecture 4

① Passenger Mix Flow Problem

- ② a Formulation
- ② b Key Path

② Column Generation Algorithm

- ② a Key Path
- ② b Implementation
- ② c Example

Passenger Mix Flow Problem

A difficult problem arises when many passengers with different itineraries compete for a limited number of seats on a single-flight segment.

Objective: to maximize passenger revenues

How? By smartly spilling passengers that are either:

- Low fare passengers
- Or that maybe recaptured in alternative itineraries

Subject to:

- The airline flight schedule
- The capacity (i.e., number of seats) per flight
- The unconstrained demand for all itineraries

This can be seen as a Multi-commodity flow problem in which:

- Passengers from a given OD paying a given fare are considered commodities
- Flight capacity is the capacity per arc

Glover, F. et all 'The Passenger-Mix Problem in the Scheduled Airlines', <https://doi.org/10.1287/inte.12.3.73>
Barnhart, C. et all 'Itinerary-Based Airline Fleet Assignment', <https://doi.org/10.1287/trsc.36.2.199.566>

Passenger Mix Flow Problem - Formulation

Sets:

L : set of flights

P : set of all passenger itineraries (paths)

P_p : set of passenger itineraries (paths) with recapture from itinerary p

Parameters:

fare_p : average fare for itinerary p

D_p : daily unconstrained demand for itinerary p

CAP_i : capacity on flight (leg) i

b_p^r : recapture rate of a pax that desires itinerary p and is allocated to r

δ_i^p : 1 if flight (leg) i belongs to the path p ; 0 otherwise

Decision Variables:

x_p^r : number of passengers from itinerary p that will travel on itinerary r

Passenger Mix Flow Problem - Formulation (2)

Objective Function:

$$\max \sum_{p \in P} \sum_{r \in P} fare_r \times x_p^r \quad (\text{OF})$$

Subject to:

$$\sum_{p \in P} \sum_{r \in P} \delta_i^r \times x_p^r \leq CAP_i, \quad \forall i \in L \quad (\text{C1}) \text{ Capacity Constraint}$$

$$\sum_{r \in P_p} \frac{x_p^r}{b_p^r} \leq D_p, \quad \forall p \in P \quad (\text{C2}) \text{ Number of passengers is lower than demand}$$

$$x_p^r \geq 0, \quad \forall p, r \in P \quad (\text{C3}) \text{ Number passengers is positive}$$

Model Size ($|F|=|\text{Flights}|$; $|P|=|\text{Itineraries}|$):

- Constraints: $|L| + |P|$
- Variables: $|P| \times |P|$

Passenger Mix Flow Problem - Keypath Formulation

Observation: Most of the passengers travel in the desired itinerary

Concept: Focus (only) on the passengers which are redirected to other itineraries

How it works:

- All passengers are assumed to follow their preferred itinerary (keypath)
- The resulting passengers assignment is often infeasible and will result in spillage
- Allocated these spilled passengers to a 'fictitious' itinerary, that would represent that these passengers will be lost
- Solve a linear programming formulation to try to reallocate these flows in alternative itineraries with spare capacity (if profitable)

Passenger Mix Flow Problem - Keypath Formulation (2)

New Notation: Focus (only) on passengers that are redirected to other itineraries

Parameters:

Q_i : daily unconstrained demand on flight (leg) i

$$Q_i = \sum_{p \in P} \delta_i^p \times D_p, \quad \forall i \in L$$

Decision Variables :

t_p^r : number of passengers that would like to travel on itinerary p and are reallocated by the airline to itinerary r

Note: the DVs from the previous formulation are linked with these DV

$$x_p^p = D_p - \sum_{r \neq p \in P} t_p^r$$

$$x_p^r = b_p^r \times t_p^r$$

Passenger Mix Flow Problem - Keypath Formulation (3)

Objective Function:

$$\min \sum_{p \in P} \sum_{r \in P} (fare_p - b_p^r \times fare_r) \times t_p^r \quad (\text{OF})$$

Subject to:

$$\sum_{p \in P} \sum_{r \in P} \delta_i^p \times t_p^r - \sum_{r \in P} \sum_{p \in P} \delta_i^p \times b_r^p \times t_r^p \geq Q_i - CAP_i, \quad \forall i \in L \quad (\text{C1}) \text{ Capacity Constraint}$$

$$\sum_{r \in P} t_p^r \leq D_p, \quad \forall p \in P \quad (\text{C2}) \text{ Number of passengers is lower than demand}$$

$$t_p^r \geq 0, \quad \forall p, r \in P \quad (\text{C3}) \text{ Number of passengers is positive}$$

Model Size ($|F|=|\text{Flights}|$; $|P|=|\text{Itineraries}|$):

- Constraints: $|L| + |P|$
- Variables: $|P| \times |P|$

Passenger Mix Flow Problem - Keypath Formulation (4)

Capacity Constraints: The amount of pax reallocated to path using flight i is less or equal than the demand spillage:

pax removed (from flight i in path p) - recaptured (for flight i in path p) \geq
demand spillage (for flight i)

$$\underbrace{\sum_{p \in P} \sum_{r \in P} \delta_i^r \times t_p^r}_{\text{pax removed from } p} - \underbrace{\sum_{r \in P} \sum_{p \in P} \delta_i^r \times b_r^p \times t_r^p}_{\text{recaptured pax from } r \text{ to } p} \geq \underbrace{Q_i - CAP_i}_{\text{demand spillage in flight } i}$$

- If there is demand spillage in i ($Q_i - CAP_i > 0$) pax need to be spilled from their keypath
- If there is capacity surplus in i ($Q_i - CAP_i < 0$) spilled pax can be recaptured to paths using flight i

Passenger Mix Flow Problem - Problem Size

This is an LP that can be solved by an LP Solver (Gurobi)

But:

- For a ‘normal’ real case problem the number of constraints and the number of decision variables explodes
 - For a network with 1000 flights ($|L| = 1000$), 5 fare-classes passengers and 5000 possible itineraries ($|P|=25000$)
 - For both formulations, this would mean:
 - 26000 constraints
 - 6.25×10^8 decision variables. If we have 8 bytes per decision variable, we need at least 800 MB
- The problem of producing an **integer** flow solution is a NP-complete problem

Outline - Lecture 4

① Passenger Mix Flow Problem

- ② a Formulation
- ② b Key Path

② Column Generation Algorithm

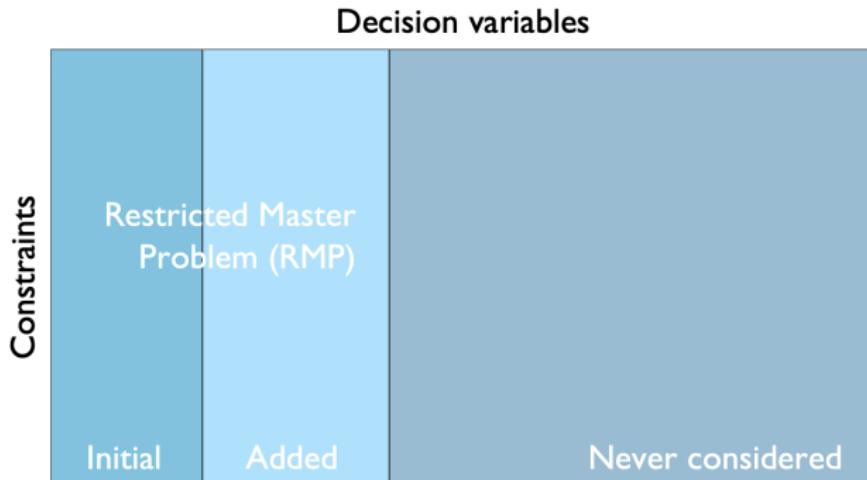
- ② a Key Path
- ② b Implementation
- ② c Example

Column Generation Algorithm

In most problems, not all decision variables (columns) are used in the final solution.

Why do we consider all these decision variables when formulating our problem?

Column generation core idea: **Start with a small (manageable) version of the problem and gradually extended it, adding new parts of the problem**



Column Generation Algorithm (2)

How it works:

- ① We initiate with an initial **set of columns** (decision variables) that makes the problem feasible: the keypath formulation – possibility to spill to the ‘fictitious’ itinerary
- ② Solving a ‘pricing problem’ and select which of the **not used columns can have higher impact** in our objective function value
 - Considering all re-allocation possibilities
 - Only the ones that reduce the lost of fare of the passengers that we are spilling or re-allocated are added to the RMP
- ③ **Add the new columns** to the problem formulation and repeat this until no more columns price out the current solution

Column Generation Algorithm - Keypath Formulation

An hypothetical case with 10 flights and 15 itineraries – If we add all columns

		Itinerary recapture (Decision Variables)																		RHS											
Constraints		(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)	(8,0)	(9,0)	(10,0)	(11,0)	(12,0)	(13,0)	(14,0)	(15,0)	(1,2)	(1,4)	(1,8)	(1,12)	(2,1)	(2,4)	(2,8)	(2,12)	(4,2)	(4,8)	(4,12)	(8,2)	(8,4)	(...)	
Flight Capacity	1	1	1	1													1-b1,2	1	1	1-b2,1	1	-b4,2	1	1	-b8,2						
	2			1	1	1	1	1									-b1,4				1	1	1	-b8,4							
	3																	-b1,8													
	4																	-b2,4													
	5																		b4,8												
	6																		-b4,12												
	7																		-b8,8												
	8																		-b8,2												
	9																		-b4,4												
	10																		-b4,12												
Itinerary demand	1	1																													
	2		1																												
	3			1																											
	4				1																										
	5					1																									
	6						1																								
	7							1																							
	8								1																						
	9									1																					
	10										1																				
	11											1																			
	12												1																		
	13													1																	
	14														1																
	15															1															
Cost Variable		fare1	fare2	fare3	fare4	fare5	fare6	fare7	fare8	fare9	fare10	fare11	fare12	fare13	fare14	fare15	
		t1,0	t2,0	t3,0	t4,0	t5,0	t6,0	t7,0	t8,0	t9,0	t10,0	t11,0	t12,0	t13,0	t14,0	t15,0	t1,2	t1,4	t1,8	t1,12	t2,1	t2,4	t2,8	t2,12	t4,2	t4,8	t4,12	t8,2	t8,4		

Note: not all recapture possibilities (DVs) are shown in this table. There could be hundreds of combinations more

Column Generation Algorithm - Keypath Formulation (2)

An hypothetical case with 10 flights and 15 itineraries – Initial iteration

Constraints		Itinerary recapture (Decision Variables)															RHS
		(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)	(8,0)	(9,0)	(10,0)	(11,0)	(12,0)	(13,0)	(14,0)	(15,0)	(..)
Flight Capacity	1	1	1														
	2			1	1	1	1										$\leq Q_1 - Cap_1$
	3							1	1	1	1						$\leq Q_2 - Cap_2$
	4											1	1	1	1		$\leq Q_3 - Cap_3$
	5	1															$\leq Q_4 - Cap_4$
	6		1	1													$\leq Q_5 - Cap_5$
	7							1									$\leq Q_6 - Cap_6$
	8												1				$\leq Q_7 - Cap_7$
	9			1	1				1				1				$\leq Q_8 - Cap_8$
	10					1	1			1				1			$\leq Q_9 - Cap_9$
Itinerary demand	1	1															$\leq Q_{10} - Cap_{10}$
	2		1														$\leq D_1$
	3			1													$\leq D_2$
	4				1												$\leq D_3$
	5					1											$\leq D_4$
	6						1										$\leq D_5$
	7							1									$\leq D_6$
	8								1								$\leq D_7$
	9									1							$\leq D_8$
	10										1						$\leq D_9$
	11											1					$\leq D_{10}$
	12												1				$\leq D_{11}$
	13													1			$\leq D_{12}$
	14														1		$\leq D_{13}$
	15																$\leq D_{14}$
Cost Variable		fare1	fare2	fare3	fare4	fare5	fare6	fare7	fare8	fare9	fare10	fare11	fare12	fare13	fare14	fare15	
		t1,0	t2,0	t3,0	t4,0	t5,0	t6,0	t7,0	t8,0	t9,0	t10,0	t11,0	t12,0	t13,0	t14,0	t15,0	

Column Generation Algorithm - Keypath Formulation (3)

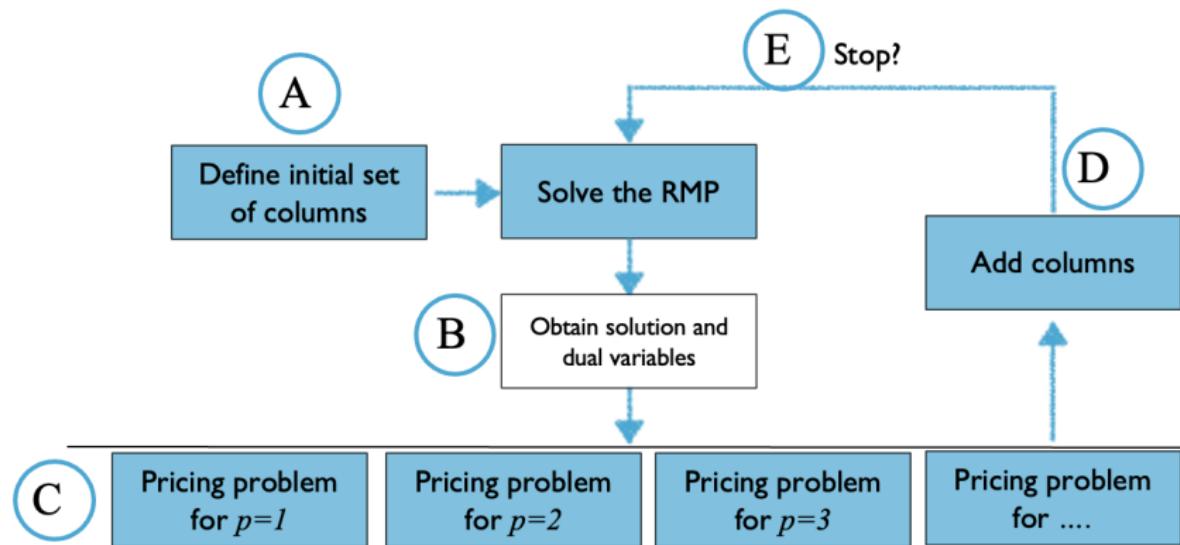
An hypothetical case with 10 flights and 15 itineraries – Second iteration

		Itinerary recapture (Decision Variables)																		
Constraints		(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)	(8,0)	(9,0)	(10,0)	(11,0)	(12,0)	(13,0)	(14,0)	(15,0)	(1,4)	(4,12)	(...)	RHS
Flight Capacity	1	1	1		1	1	1	1									-b1,4	1		≥ Q1 - Cap1
	2																		≥ Q2 - Cap2	
	3																		≥ Q3 - Cap3	
	4																		≥ Q4 - Cap4	
	5	1																	≥ Q5 - Cap5	
	6		1	1															≥ Q6 - Cap6	
	7								1										≥ Q7 - Cap7	
	8																		≥ Q8 - Cap8	
	9			1	1					1									≥ Q9 - Cap9	
	10					1	1				1								≥ Q10 - Cap10	
Itinerary demand	1	1																	≤ D1	
	2		1																≤ D2	
	3			1															≤ D3	
	4				1														≤ D4	
	5					1													≤ D5	
	6						1												≤ D6	
	7							1											≤ D7	
	8								1										≤ D8	
	9									1									≤ D9	
	10										1								≤ D10	
	11											1							≤ D11	
	12												1						≤ D12	
	13													1					≤ D13	
	14														1				≤ D14	
	15																		≤ D15	
Cost Variable	fare1	fare2	fare3	fare4	fare5	fare6	fare7	fare8	fare9	fare10	fare11	fare12	fare13	fare14	fare15		...			
	t1,0	t2,0	t3,0	t4,0	t5,0	t6,0	t7,0	t8,0	t9,0	t10,0	t11,0	t12,0	t13,0	t14,0	t15,0		...	t1,4		
																		...	t4,12	

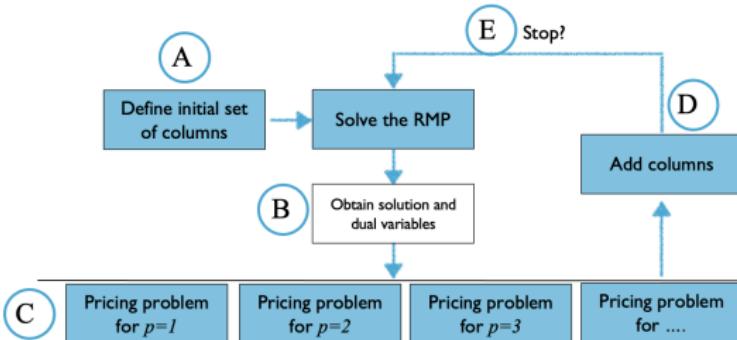
Column Generation Algorithm - Implementation

How to implement the algorithm?

- ① Solve the *Restricted Master Problem (RMP)*
- ② Solve the *Pricing Problem* to see if any columns need to be added to the RMP
- ③ If columns identified, add them and return to 1; otherwise stop

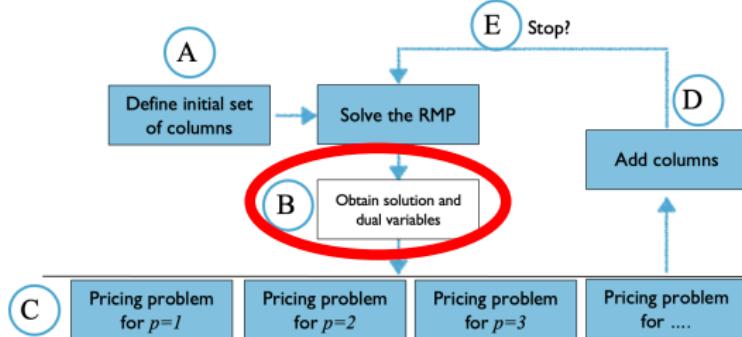


Column Generation Algorithm - Implementation (2)



- Let's assume that:
 - There is a 'fictitious' itinerary 0 with $fare_0 = 0$
 - $b_p^0 = 1$ for all p (i.e., the recapture rate to the 'fictitious' itinerary)
 - $b_p^p = 1$ for all p (i.e., the recapture rate to itinerary itself)
- Forget the 'keypath' columns (t_p^p) - they will, by default, be the option selected
- Start with the 'fictitious' itinerary t_p^0 as the only option for all itineraries which have a lack of capacity in their flights. This will work as a buffer for those itineraries p with passengers being spilled due to the flight capacity

Column Generation Algorithm - Implementation (3)



- What are the dual variables?
 - They are the solution to the dual problem
 - They are related to each constraint of our primal problem

Objective Function: $\min \sum_{p \in P} \sum_{r \in P} (fare_p - b_p^r \times fare_r) \times t_p^r$

Dual variables:

Subject to:

$$\sum_{p \in P} \sum_{r \in P} \delta_i^r \times t_p^r - \sum_{r \in P} \sum_{p \in P} \delta_i^r \times b_r^P \times t_r^P \geq Q_i - CAP_i, \forall i \in L \quad \rightarrow \quad \pi_i$$

$$\sum_{r \in P_p} t_p^r \leq D_p, \forall p \in P \quad \rightarrow \quad \sigma^P$$

Note: σ is always non-positive because it's a \leq constraint in a minimisation problem

Column Generation Algorithm - Implementation (4)

Recall the Simplex Method & Dual Theory:

Every LP problem (known as the primal) has its dual. If the primal is maximization, then the dual is minimization, and vice-versa. The solutions for the dual problem are the same as for the primal

The simplex method is applied simultaneously to its dual problem. The two methods move in complete sequence, obtaining feasible solutions to the dual problems with each iteration

- The objective function of the dual problem is to maximize the cost of reallocation
- There is one dual variable for each explicit constraint in the primal
- The meaning of the dual variables π_i and σ^P are reallocation costs
- The dual problem's variables are often referred to as 'shadow prices'
- The dual variables represent the marginal effect on the primal objective per unit change in each primal constraint limit - Increasing (decreasing) the fair by a small amount will reduce (increase) the total amount of reallocated passengers

Column Generation Algorithm - Implementation (5)

Slack Variables:

Additional variables that are introduced into the linear constraints of a linear program to transform them from inequality constraints to equality constraints. They are needed in the constraints to transform them into solvable equalities with one definite answer

After the slack variables are introduced, the tableau can be set up to check for optimality

$$x_1 + 3x_2 + 2x_3 \leq 10 \Rightarrow x_1 + 3x_2 + 2x_3 + s_1 = 10$$

$$-x_1 - 5x_2 - x_3 \geq -8 \Rightarrow x_1 + 5x_2 + x_3 + s_2 = 8$$

$$x_1, x_2, x_3 \geq 0 \Rightarrow x_1, x_2, x_3, s_1, s_2 \geq 0$$

Complementary Slackness Property:

If the value of the dual variable (shadow price) associated with a constraint is nonzero, then that constraint must be satisfied with equality. Further, if a constraint is satisfied with strict inequality, then its corresponding dual variable must be zero

Column Generation Algorithm - Implementation (6)

The Pricing Problem:

Calculate marginal values of a unit of each 'resource' (required minimum, total capital). The constraints ensure that your prices sufficiently explain the primal objective; the dual objective (minimization) prevents you from 'overpricing' resources

To estimate if there 'slackness' to improve the objective function value (z_j):

$$\text{slack} = z_j - c_j = \sum_{i \in N} a_{ij} \times y_i - c_{ij}$$

where:

- c_j is the cost of activity j
- a_{ij} is amount of resource i used by activity j
- y_i is the dual variable associated with the limited resource i
- c_{ij} is the cost of using resource i for activity j

If there is, at least, one constraint with a slack lower than zero, the respective primal decision variable should be added to the base

Otherwise, the optimal solution was founded!

Column Generation Algorithm - Implementation (7)

Objective Function:

$$\min \sum_{p \in P} \sum_{r \in P} (fare_p - b_p^r \times fare_r) \times t_p^r \quad (\text{OF})$$

Subject to:

$$\sum_{p \in P} \sum_{r \in P} \delta_i^p \times t_p^r - \sum_{r \in P} \sum_{p \in P} \delta_i^p \times b_r^p \times t_r^p \geq Q_i - CAP_i, \quad \forall i \in L \quad (\text{C1}) \text{ Capacity Constraint}$$

$$\sum_{r \in P} t_p^r \leq D_p, \quad \forall p \in P \quad (\text{C2}) \text{ Number of passengers is lower than demand}$$

Then, for all combinations p and $r \in P_p$:

$$c_j' = \sum_{i \in N} a_{ij} \times y_i - c_{ij} \Rightarrow c_p^{r'} = \sum_{i \in L} (\delta_i^p - \delta_i^r \times b_p^r) \times \pi_i + \sigma_p - (fare_p - b_p^r \times fare_r)$$

where $c_p^{r'}$ is the slackness of decision variable t_p^r

Column Generation Algorithm - Implementation (7)

Objective Function:

$$\min \sum_{p \in P} \sum_{r \in P} (fare_p - b_p^r \times fare_r) \times t_p^r \quad (\text{OF})$$

Subject to:

$$\sum_{p \in P} \sum_{r \in P} \delta_i^p \times t_p^r - \sum_{r \in P} \sum_{p \in P} \delta_i^p \times b_r^p \times t_r^p \geq Q_i - CAP_i, \quad \forall i \in L \quad (\text{C1}) \text{ Capacity Constraint}$$

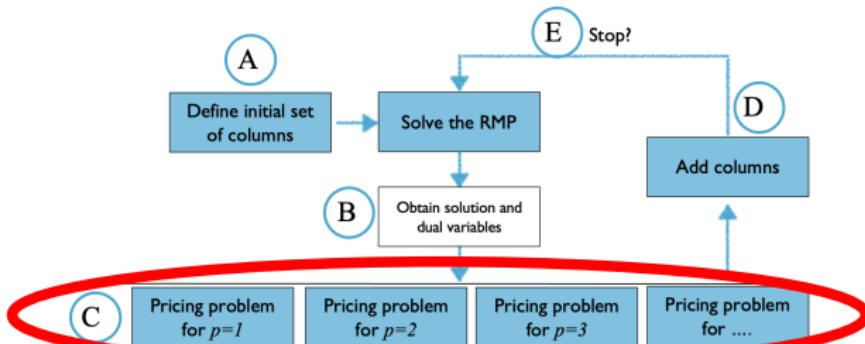
$$\sum_{r \in P} t_p^r \leq D_p, \quad \forall p \in P \quad (\text{C2}) \text{ Number of passengers is lower than demand}$$

Then, for all combinations p and $r \in P_p$:

$$c_j' = \sum_{i \in N} a_{ij} \times y_i - c_{ij} \Rightarrow c_p^{r'} = \sum_{i \in L} (\delta_i^p - \delta_i^r \times b_p^r) \times \pi_i + \sigma_p - (fare_p - b_p^r \times fare_r)$$

where $c_p^{r'}$ is the slackness of decision variable t_p^r

Column Generation Algorithm - Implementation (8)



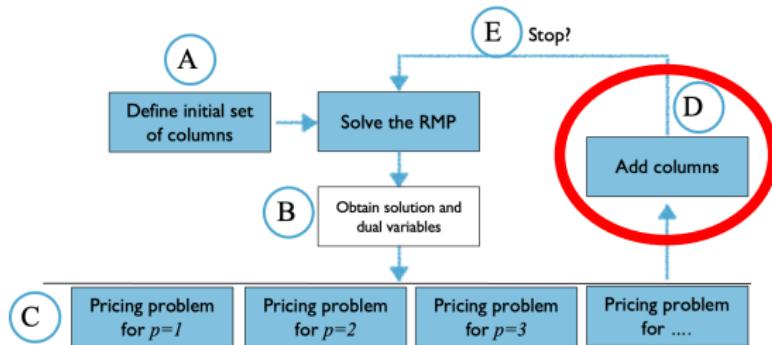
The slackness can be rewritten in the following way:

$$c_p^{r'} = \text{fare}_p - b_p^r \times \text{fare}_r - \sum_{i \in p} \pi_i + b_p^r \sum_{j \in r} \pi_j - \sigma_p$$

or

$$c_p^{r'} = \underbrace{\left(\text{fare}_p - \sum_{i \in p} \pi_i \right)}_{\text{modified fare in itinerary } p} - b_p^r \times \underbrace{\left(\text{fare}_r - \sum_{j \in r} \pi_j \right)}_{\text{modified fare in the alternative itinerary } r} - \underbrace{\sigma_p}_{\text{additional value for transporting an additional passenger from } p}$$

Column Generation Algorithm - Implementation (9)



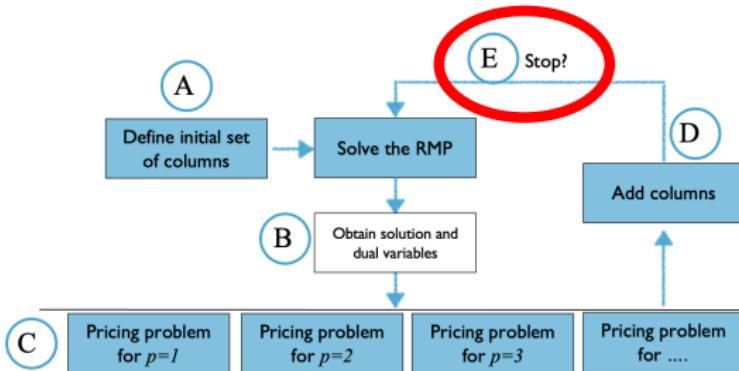
- For every possible combination p to r , we solve the pricing problem
- If slackness is negative, i.e.:

$$\left(fare_p - \sum_{i \in p} \pi_i \right) - b_p^r \times \left(fare_r - \sum_{j \in r} \pi_j \right) - \sigma_p < 0$$

this column (option to accommodate passengers from p in r) 'prices out' current solution - i.e., spill costs are reduced if added to the RMP —→ **add column t_p^r**

- Otherwise, this column (alternative re-accommodation option) does not reduce the spill costs —→ **don't add**

Column Generation Algorithm - Implementation (10)



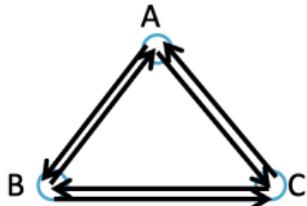
- If:
 - None of the non-added re-accommodation options 'price out' existing solution
 - None of the constraints (from the primal/original problem) is violated
 - If we have non-null dual variables they are from (aka, complementary slackness verification)
 - Flights with the capacity used up to the limit
 - Itineraries in which all passengers are re-accommodated
- Then the solution obtained with the current columns is optimal and we can **stop** adding more columns

Column Generation Algorithm - Example

Problem with 3 airports, 10 flights and 15 itineraries. There are recapture options between 17 combinations of itineraries.

Flights:

#	O	D	Departure Time	Ready Time	Cap
102	A	B	08:30:00	10:30:00	108
301	C	B	09:00:00	11:00:00	144
201	B	C	11:00:00	13:00:00	144
104	A	B	11:30:00	13:30:00	108
202	B	A	14:00:00	16:15:00	108
302	C	A	14:15:00	16:45:00	108
303	C	B	16:30:00	18:00:00	144
204	B	A	17:20:00	19:45:00	108
203	B	C	17:30:00	19:30:00	108
101	A	C	18:10:00	19:40:00	108



Recapture possibilities:

Itineraries:

#	Origin	Destination	Demand	Fare	Leg1	Leg2
1	A	B	90	\$ 150	102	
2	A	B	120	\$ 170	104	
3	B	A	70	\$ 80	202	
4	B	A	100	\$ 70	204	
5	B	A	50	\$ 50	201	302
6	A	C	120	\$ 120	101	
7	A	C	80	\$ 100	102	201
8	A	C	70	\$ 100	104	203
9	C	A	140	\$ 140	302	
10	C	A	125	\$ 80	301	202
11	B	C	50	\$ 150	201	
12	B	C	180	\$ 180	203	
13	B	C	30	\$ 100	202	101
14	C	B	75	\$ 75	301	
15	C	B	80	\$ 80	303	

#	From	To	Rate
1	1	2	0,3
2	2	1	0,1
3	3	4	0,2
4	4	5	0,2
5	4	3	0,1
6	3	5	0,1
7	6	7	0,3
8	7	8	0,2
9	6	8	0,1

#	From	To	Rate
10	9	10	0,2
11	10	9	0,1
12	11	12	0,4
13	12	13	0,3
14	11	13	0,2
15	12	11	0,1
16	14	15	0,2
17	15	14	0,1

Column Generation Algorithm - Example (1)

- Unconstrained demand per flight:
 - Flight 102: 170 (Cap = 108) $\rightarrow Q_i - CAP_i = 62$
 - Flight 301: 200 (Cap = 144) $\rightarrow Q_i - CAP_i = 56$
 - Flight 201: 180 (Cap = 144) $\rightarrow Q_i - CAP_i = 36$
 - ...
 - Flight 303: 80 (Cap = 144) $\rightarrow Q_i - CAP_i = -64$
 - Flight 204: 100 (Cap = 108) $\rightarrow Q_i - CAP_i = -8$
 - ...
- Initiate the RMP only with the ‘fictitious’ itinerary (index = 0)

Column Generation Algorithm - Example (2)

RMP - Iteration 1:

Objective Function:

$$\min \sum_{p \in P} \sum_{r \in P} (fare_p - \overbrace{b_p^r \times fare_r}^{fare_0 = 0}) \times t_p^r$$

$$\min : 150t_1^0 + 170t_2^0 + 80t_3^0 + 70t_4^0 + 50t_5^0 + 120t_6^0 + \dots + 80t_{15}^0$$

Subject to:

- Flight 102: $t_1^0 + t_7^0 \geq 62$
- Flight 301: $t_{10}^0 + t_{14}^0 \geq 56$
- Flight 201: $t_5^0 + t_7^0 + t_{11}^0 \geq 36$
- Flight 104: $t_2^0 + t_8^0 \geq 82$
- ...

Column Generation Algorithm - Example (3)

RMP - Iteration 1:

- OF Value = 46580

Decision variables (non null): Dual variables (non null):

- | | |
|-------------------|----------------------|
| • $t_2^0 = 12$ | $\pi_{102} = 100$ |
| • $t_5^0 = 50$ | $\pi_{104} = 170$ |
| • $t_6^0 = 12$ | $\pi_{202} = 80$ |
| • $t_7^0 = 62$ | $\pi_{302} = 140$ |
| • $t_8^0 = 70$ | $\pi_{203} = 180$ |
| • $t_9^0 = 32$ | $\pi_{101} = 120$ |
| • $t_{10}^0 = 87$ | $\sigma_5 = -90$ |
| • $t_{12}^0 = 72$ | $\sigma_8 = -250$ |
| • $t_{13}^0 = 30$ | $\sigma_{13} = -100$ |

$$c_p^{r'} = \left(fare_p - \sum_{i \in p} \pi_i \right) - b_p^r \times \left(fare_r - \sum_{j \in r} \pi_j \right) - \sigma_p$$



$$c_1^2 = (150 - 100) - 0.3(170 - 170) - 0 = 50$$

$$c_2^1 = (170 - 170) - 0.1(150 - 100) - 0 = -5$$

$$c_3^4 = (80 - 80) - 0.2(70 - 0) - 0 = -14$$

$$c_4^5 = (70 - 0) - 0.2(50 - 0 - 140) - 0 = 88$$

$$c_4^3 = (70 - 0) - 0.1(80 - 80) - 0 = 70$$

...

Add columns: $t_2^1, t_3^4, t_{12}^{11}$

Column Generation Algorithm - Example (4)

RMP - Iteration 2:

Objective Function:

$$\min \sum_{p \in P} \sum_{r \in P} (fare_p - b_p^r \times fare_r) \times t_p^r$$

$$\begin{aligned} \min : & 150t_1^0 + 170t_2^0 + 80t_3^0 + \dots + 80t_{15}^0 \\ & + (170 - 0.1 \times 150)t_2^1 + (80 - 0.2 \times 70)t_3^4 + (180 - 0.1 \times 150)t_{12}^{11} \end{aligned}$$

Subject to:

- Flight 102: $t_1^0 + t_7^0 - 0.1t_2^1 \geq 62$
- Flight 301: $t_{10}^0 + t_{14}^0 \geq 56$
- Flight 201: $t_5^0 + t_7^0 + t_{11}^0 - 0.1t_{12}^{11} \geq 36$
- Flight 104: $t_2^0 + t_8^0 + t_2^1 \geq 82$
- ...

Column Generation Algorithm - Example (5)

RMP - Iteration 2:

- OF Value = 45006 (-1574)

Decision variables

(non null):

- $t_2^1 = 12$
- $t_3^4 = 31$
- $t_5^0 = 50$
- $t_6^0 = 12$
- $t_7^0 = 63.2$
- $t_8^0 = 70$
- $t_9^0 = 32$
- $t_{10}^0 = 56$
- $t_{12}^{11} = 72$
- $t_{13}^0 = 30$

$$b_3^4 = 0.2$$

6.2 pax to itinerary 4
24.8 pax spilled

Dual variables

(non null):

$$\begin{aligned}\pi_{102} &= 100 \\ \pi_{301} &= 14 \\ \pi_{104} &= 165 \\ \pi_{202} &= 66 \\ \pi_{302} &= 140 \\ \pi_{203} &= 165 \\ \pi_{101} &= 120 \\ \sigma_5 &= -90 \\ \sigma_8 &= -230 \\ \sigma_{13} &= -86\end{aligned}$$

No more columns to be added → Optimal solution

Airline Planning and Optimisation - AE4423-20

Lecture 5

Schedule Design & Fleet Assignment

Marta Ribeiro
Assistant Professor
Air Transport & Operation
Control & Operations

Delft University of Technology, The Netherlands

November, 2024

Course Information - Content

Management

Planning structure
Performance Indicators
Demand and Supply
Market share

Tactical Planning

Passenger Mix Flow
Fleet Assignment
Aircraft Rotation
Crew Scheduling
Maintenance



Strategical Planing

Network Structure
Demand Forecast
Network Planning
Fleet Planning

Operations Research

MILP Models

Multi-commodity Flow Problems
Shortest-path Algorithms
Column Generation Algorithm
Dynamic Programming

Outline - Lecture 5

① Schedule Development

- ① Schedule Development
- ② The Fleet Assignment Problem
- ③ Time-space Networks
- ④ Example
- ⑤ Fleet Assignment Model (FAM)
- ⑥ Example Solution

② Itinerary-based Fleet Assignment Problem

- ① Model
- ② Solution Technique

③ Timetable Design

- ① Complexity and Challenge
- ② FAM adaptations

Schedule Development

Given a set of routes to be operated in an airline network, a fleet of aircraft, and a pool of crew members, the schedule development involves four different but interrelated tasks:

Today

Timetable design: how often should the airline operate flights on the selected route(s) and at what times should flights depart?

Fleet assignment: what type of aircraft should be used to operate each flight in our timetable?

Next Lectures

Crew scheduling: how to assign crew (cabin & pilots) to each flight guaranteeing the operation of the flights and satisfying all work rules

Aircraft rotation: how should each specific aircraft (tail number) be flown over the airline's network in order to ensure a consistent operation?

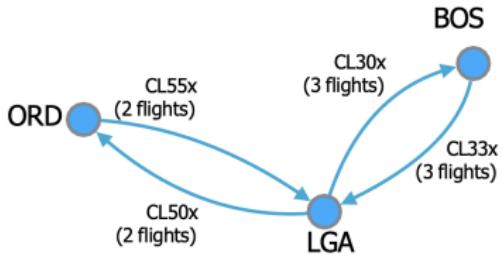
Schedule Development (2)

The process begins a year or more in advance and continues until actual departure time:

- Fleet plans established first, based on forecasted demands - typically made **2-5 years earlier**
- Route evaluation is done in parallel but final decisions are made later - **1-2 years before operation**
- Timetables and fleet allocations defined **12-6 months in advance**
- Crew assignment cover the activities to be executed by crew member over a period of, typically, **1-2 months in advance**
- Aircraft rotation is done in parallel with maintenance planning and usually is defined **1-2 weeks in advance**
- Final revisions and disruptions managed until the flight departs

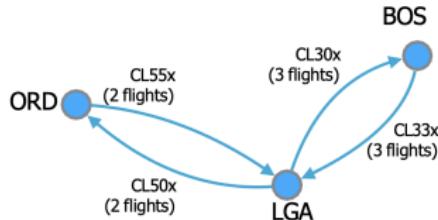
The Fleet Assignment Problem

- Flight assignment is a **tactical decision** made after an initial timetable has been developed
- The **objective** is to minimise the operation costs, to maximise profits or to reduce the costs of the demand not served (i.e., spilled). Given the fleet, the flights in the timetable, and the demand per flight
- Many airlines have fleet assignment models on large-scale mathematical network optimisation methods (Belobaba et al., 2007):



AC Type	No.	Seats	Op. Cost per Flight	
			LGA-BOS	LGA-ORD
DC9	1	120	10000	15000
B737	2	150	12000	17000
A300	2	250	15000	20000

The Fleet Assignment Problem (2)



Flight no.	From	To	Departure Time (EST)	Arrival Time (EST)	Fare	Demand (Passengers)
CL301	LGA	BOS	11:00	12:00	150	250
CL302	LGA	BOS	12:00	13:00	150	250
CL303	LGA	BOS	14:00	15:00	150	100
CL331	BOS	LGA	08:00	09:00	150	150
CL332	BOS	LGA	11:30	12:30	150	300
CL333	BOS	LGA	14:00	15:00	150	150
CL501	LGA	ORD	12:00	15:00	400	150
CL502	LGA	ORD	13:00	16:00	400	200
CL551	ORD	LGA	08:00	11:00	400	200
CL552	ORD	LGA	09:30	12:30	400	150

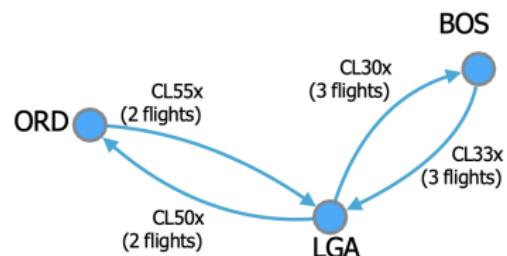
What is the optimal fleet assignment solution? (Objective: maximise profitability)

The Fleet Assignment Problem (3)

A 'greedy' solution which assigns the AC type that maximises profit for each flight leg results in:

Flight no.	DC9	B737	A300
CL301	8000	10500	22500
CL302	8000	10500	22500
CL303	5000	3000	0
CL331	8000	10500	7500
CL332	8000	10500	22500
CL333	8000	10500	7500
CL501	33000	43000	40000
CL502	33000	43000	60000
CL551	33000	43000	60000
CL552	33000	43000	40000

(=min(Capacity, Demand)*Fare - Op. Cost)



A300s:

- CL301
- CL302
- CL332
- CL502
- CL551

B737s:

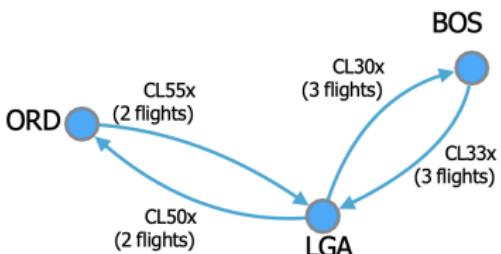
- CL331
- CL333
- CL501
- CL552

DC9s:

- CL303

What is the problem with this solution?

The Fleet Assignment Problem (4)



Flight no.	From	To	Departure Time (EST)	Arrival Time (EST)	Fare	Demand (Passengers)
A300	CL301	LGA	BOS	11:00	12:00!	150 250
A300	CL302	LGA	BOS	12:00	13:00	150 250
DC10	CL303	LGA	BOS	14:00	15:00	150 100
B737	CL331	BOS	LGA	08:00	09:00	150 150
A300	CL332	BOS	LGA	11:30!	12:30	150 300
B737	CL333	BOS	LGA	14:00	15:00!	150 150
B737	CL501	LGA	ORD	12:00!	15:00	400 150
A300	CL502	LGA	ORD	13:00	16:00	400 200
A300	CL551	ORD	LGA	08:00	11:00	400 200
B737	CL552	ORD	LGA	09:30	12:30	400 150

DC10

What to do with this AC?

A300

2 starting in LGA

1 starting in ORD

there are only 2 AC

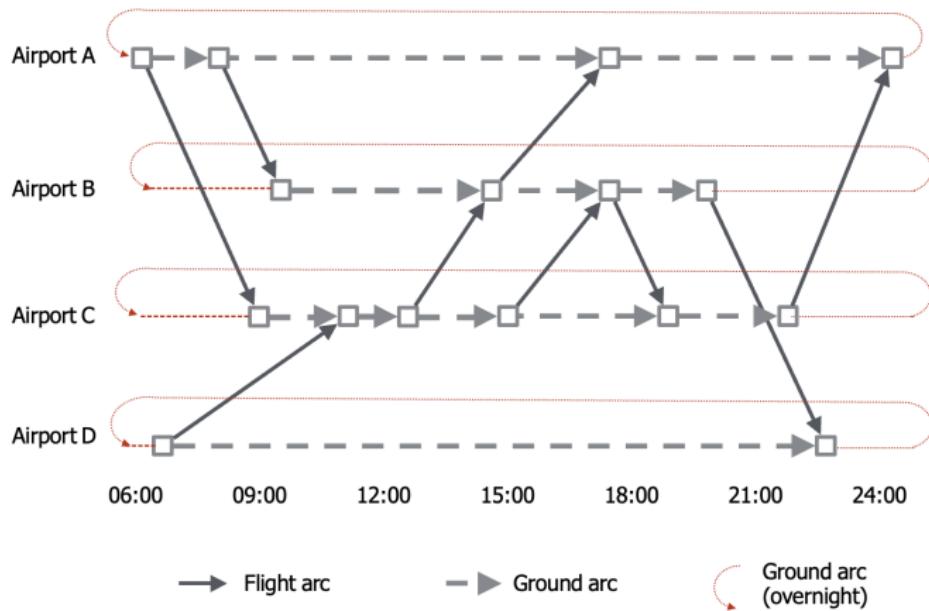
B737

2 flights from BOS to LGA non from LGA to BOS

How can we avoid this?

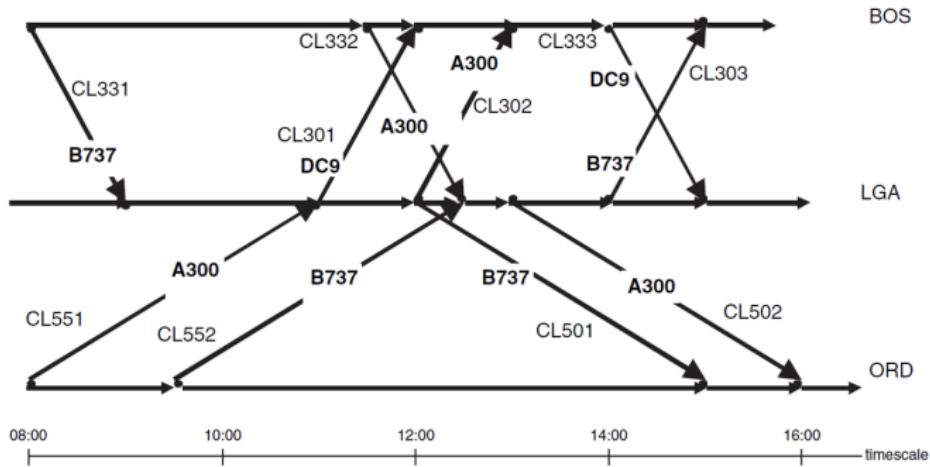
Time-space Networks

To capture the temporal nature of this problem, flight networks are modelled using **time-space networks**:



Time-Space Networks (2)

The main differences between a time-space networks and a static representation is: *time-space networks are used to model fleet allocation alternatives, while a schedule map already represents one solution.*

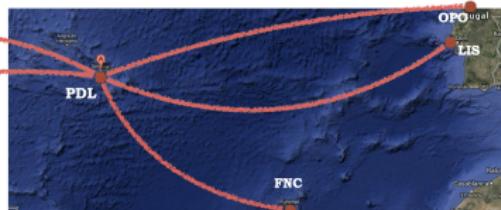


Static network with the solution to the previous problem (Source: Belobaba et al., 2009)

Time-Space Networks - Example

Atlantic Airlines operates flights from its hub in Ponta Delgada (Portugal). It has 4 A310's and 4 A320's and, in a normal day operates the 18 flights. Determine the aircraft that should operate each flight in order to minimise costs of operation.

Flight no.	From	To	Departure Time (EST)	Arrival Time (EST)	Demand (Pax)
S4120	PDL	LIS	09:25	11:30	195
S4220	PDL	LIS	11:20	13:25	167
S4124	PDL	LIS	16:05	18:10	151
S4128	PDL	LIS	22:05	00:10	184
S4320	PDL	OPO	09:35	11:45	166
S4212	PDL	OPO	19:00	21:10	134
S4160	PDL	FNC	10:35	12:45	155
S4222	PDL	YTO	17:10	23:45	214
S4221	PDL	BOS	16:00	21:45	198
S4121	LIS	PDL	06:30	08:45	148
S422A	LIS	PDL	08:05	10:30	191
S4125	LIS	PDL	11:00	14:35	154
S4129	LIS	PDL	18:50	21:05	194
S4321	OPO	PDL	13:30	15:50	166
S4213	OPO	PDL	19:05	21:25	134
S4161	FNC	PDL	17:05	19:10	155
S4223	YTO	PDL	02:45	08:20	214
S422B	BOS	PDL	03:15	08:00	189



AC Type	Fleet	Seats	Range (km)	CASK (cent\$/ASK)	TAT (min)
A310	4	304	9600	5.3	40
A320	4	164	4400	4.6	25

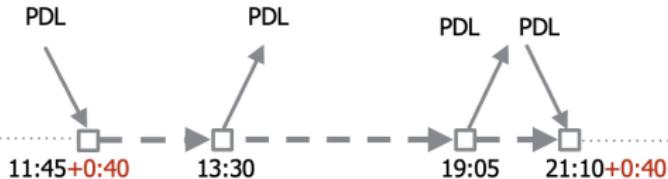
LIS OPO FNC YTO BOS

Distance From PDL 1449 1509 985 4496 3854

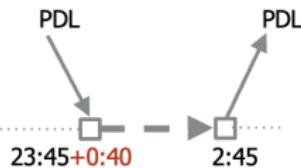
Time-Space Networks - Example (2)

Step 1/2

Airport: OPO
Type: **A310**



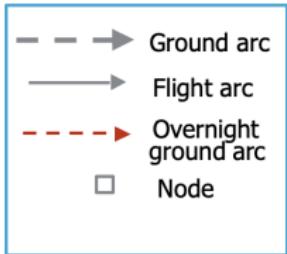
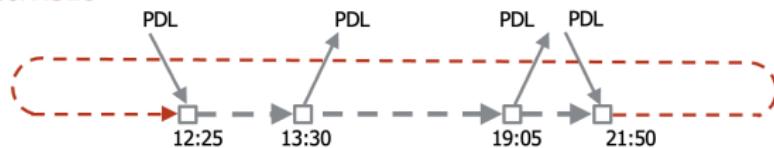
Airport: YTO
Type: **A310**



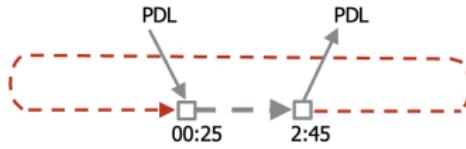
Time-space Networks - Example (3)

Airport: OPO
Type: A310

Step 2/2



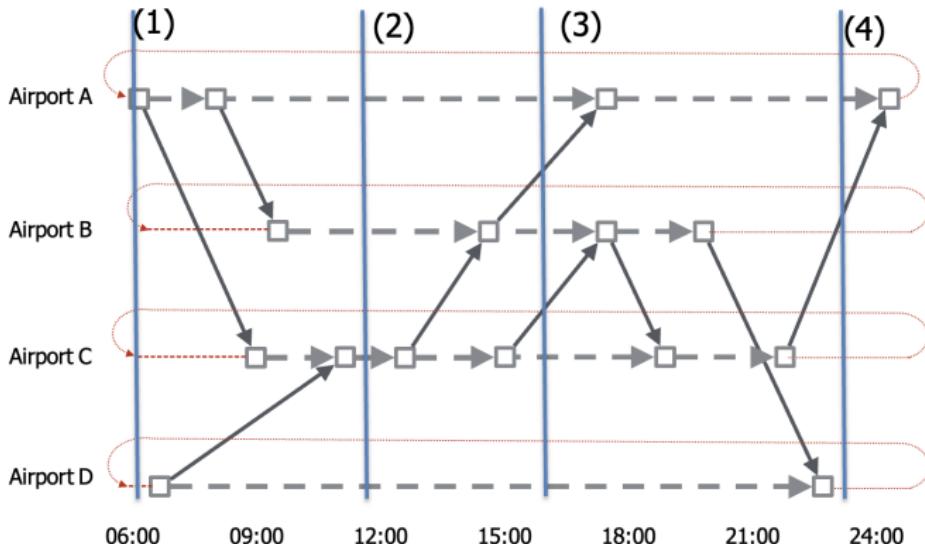
Airport: YTO
Type: A310



time

The Fleet Assignment Model

How to know the number of aircraft used in a time-space network?



Number of aircraft in the arcs crossing (1), (2), (3) and (4) are the same and it tell us the number of aircraft being used

The Fleet Assignment Model (2)

Objective Function: Costs minimisation (OF)

Subject to:

- Flight schedule coverage (C1)
Each flight covered exactly once by one fleet type
- Balance at each node (time and space) (C2)
The number of AC arriving = AC departing, for each type
- Number of aircraft available (C3)
Cannot assign more aircraft than the ones available, for each type
- Other Restrictions (C4)
Maintenance, Range, number of airport stands, etc

The Fleet Assignment Model (3)

Sets:	N	: set of airports
	F	: set of flights
	K	: set of aircraft types
	G^k	: set of ground arcs
	NG^k	: set of flight and ground arcs intercept by the time cut
	$O(k, n)$: flights arcs originating at node n in fleet k
	$I(k, n)$: flights arcs terminating at node n in fleet k
	n^+	: ground arcs originating at any node n
	n^-	: ground arcs terminating at any node n
 Parameters:	d_i	: distance of flight i
	s	: number of seats per aircraft
	rev	: revenue per Revenue Passenger Kilometers (RPK) flown (average yield)
	q_i	: traffic demand in flight i
	R^k	: range of AC type k
	AC^k	: number of aircraft in the fleet of type k
	oc_i^k	: unit operation cost (/ASK) for AC type k for flight i
 Decision Variables:	f_i^k	: 1 if flight arc i is assigned to aircraft type k , 0 otherwise
	y_a^k	: number of aircraft of type k on the ground arc a (integer)

The Fleet Assignment Model (4)

Objective Function:

$$\min \sum_{i \in F} \sum_{k \in K} oc_i^k \times s^k \times d_i \times f_i^k \quad (+ \text{ Ground arc costs}) \quad (\text{OF})$$

Subject to:

$$\sum_{k \in K} f_i^k = 1, \quad \forall i \in F \quad (\text{C1})$$

$$y_{n^+}^k + \sum_{i \in O(k, n)} f_i^k - y_{n^-}^k - \sum_{i \in I(k, n)} f_i^k = 0, \quad \forall n \in N^k, k \in K \quad (\text{C2})$$

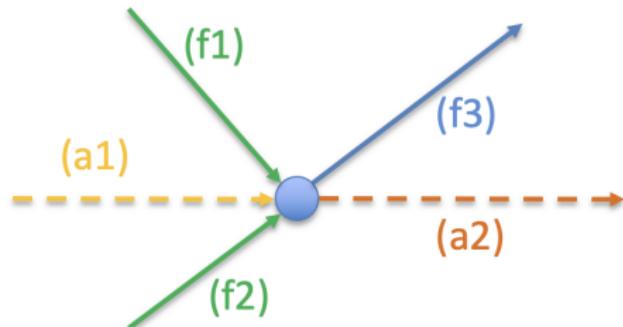
$$\sum_{a \in NG^k} y_a^k + f_a^k \leq AC^k, \quad \forall k \in K \quad (\text{C3})$$

$$f_i^k \in \{0, 1\}, \quad \forall k \in K$$

$$y_a^k \geq 0, \quad \forall a \in G^k, k \in K$$

The Fleet Assignment Model (5)

How do constraints C2 work?



$$\underbrace{y_{n^+}^k}_{a2} + \underbrace{\sum_{i \in O(k,n)} f_i^k}_{f3} - \underbrace{y_n^k}_{a1} - \underbrace{\sum_{i \in I(k,n)} f_i^k}_{f_1+f_2} = 0, \quad \forall i \in n \in N^k, k \in K$$

||

$$y_{a2}^k + (f_{f3}^k) - y_{a2}^k - (f_{f1}^k + f_{f2}^k) = 0$$

The Fleet Assignment Model - Example Solution

No fleet constraint:

- A310:
Flights: S4222, S4223
- AC grounded overnight:

	A310	A320
PDL	0	2
LIS	0	3
OPO	0	1
FNC	0	0
YTO	1	0
BOS	0	1

- Cost = $351 (10^3 \text{ €})$

$4 \times \text{A310} + 4 \times \text{A320}$:

- A310:
Flights: S4120, S4220, S4222, S4231, S4223
- AC grounded overnight:

	A310	A320
PDL	1	1
LIS	0	2
OPO	0	1
FNC	0	0
YTO	1	0
BOS	0	1

- Cost = $389 (10^3 \text{ €})$

Outline - Lecture 5

① Schedule Development

- ① Schedule Development
- ② The Fleet Assignment Problem
- ③ Time-space Networks
- ④ Example
- ⑤ Fleet Assignment Model (FAM)
- ⑥ Example Solution

② Itinerary-based Fleet Assignment Problem

- ① Model
- ② Solution Technique

③ Timetable Design

- ① Complexity and Challenge
- ② FAM adaptations

Itinerary-based Fleet Assignment Problem

Itinerary-based Fleet Assignment Model (IFAM) = Passenger Mix Flow (PMF) + Fleet Assignment Model (FAM)

IFAM integrates the standard FAM with the PMF model:

- The PMF simulates spill and recapture effects
- The FAM simulates the capacity allocation to the network

The formulation is a combination of both previous models:

- Associate costs of assigning capacity (aircraft types) to flights
- Associate revenues with passenger flow variables (including spill and recovery)

The variable definitions are the same, except that CAP_i is now replaced by $s_k \times f_i^k$, where s_k is the capacity (number of seats) of fleet type k

Proposed in: Barnhart et al (2002) 'Itinerary-Based Airline Fleet Assignment', Transportation Science, 36(2):199–217

Itinerary-based Fleet Assignment Problem (2)

Problem (re-)definition:

The IFAM problem consists on:

- Objective: maximize fleet allocation contribution (i.e., reduce costs)
- How: assigning aircraft types to scheduled flights in order to
 - Better match demand per flight with the number of supplied seats
 - **And capturing the network effects for spillage and recapture**
- Subject to:
 - The airline flight schedule (each flight needs to be flown)
 - Conservation flow balance at the nodes
 - Limited number of aircraft per type in the fleet
 - Turn-around times at each station
 - **Unconstrained passenger demands by itinerary**
 - **Recapture rates between itineraries**

Itinerary-based Fleet Assignment Problem - Model

Objective Function:

$$\min \sum_{i \in L} \sum_{k \in K} c_{k,i} \times f_i^k + \sum_{p \in P} \sum_{r \in P} (fare_p - b_p^r \times fare_r) \times t_p^r \quad (\text{OF})$$

Subject to:

$$\sum_{k \in K} f_i^k = 1, \quad \forall i \in L \quad (\text{C1})$$

$$y_{n^+}^k + \sum_{i \in O(k,n)} f_i^k - y_n^k - \sum_{i \in I(k,n)} f_i^k = 0, \quad \forall n \in N^k, k \in K \quad (\text{C2})$$

$$\sum_{a \in NG^k} y_a^k + f_a^k \leq AC^k, \quad \forall k \in K \quad (\text{C3})$$

$$\sum_{k \in K} s_k \times f_i^k + \sum_{p \in P} \sum_{r \in P} \delta_i^p \times t_p^r - \sum_{r \in P} \sum_{p \in P} \delta_i^p \times b_r^p \times t_r^p \geq Q_i, \quad \forall i \in L \quad (\text{C4})$$

PMF

$$\sum_{r \in P} t_p^r \leq D_p, \quad \forall p \in P \quad (\text{C5})$$

$$f_i^k \in \{0, 1\}, \quad \forall k \in K$$

$$y_n^k \geq 0, \quad \forall n \in G^k, k \in K$$

$$t_p^r \geq 0, \quad \forall p, r \in P$$

Itinerary-based Fleet Assignment Problem - Solution Technique

- ① Consider the LP relaxed version of the MILP model presented in the previous slide (i.e., f_i^k and y_a^k assumed continuous)
- ② Follow the same procedure described for the PMF problem
 - Start IFAM with recapture from other itineraries to the 'fictitious' one
 - Column generation - add recaptures that 'price out'
 - Resolve IFAM until columns generation adds columns
- ③ Once all rows and columns are generated:
 - Reassume the MILP (i.e., $f_i^k = \{0, 1\}$)
 - Solve using branch & bound (note: using the LP Solver)

Outline - Lecture 5

① Schedule Development

- ① Schedule Development
- ② The Fleet Assignment Problem
- ③ Time-space Networks
- ④ Example
- ⑤ Fleet Assignment Model (FAM)
- ⑥ Example Solution

② Itinerary-based Fleet Assignment Problem

- ① Model
- ② Solution Technique

③ Timetable Design

- ① Complexity and Challenge
- ② FAM adaptations

Timetable Design - Complexity and Challenge

Why does the timetable design usually follows an **incremental approach**?

- ① Building a new timetable from scratch requires good quality demand data which is rarely available
- ② Building an entirely new full schedule is computationally difficult (if not intractable)
- ③ Frequency changes require investment changes at airport stations (e.g., gates, slots, check-in counters)
- ④ Airlines develop a fidelity bound with their customers, especially in business markets in which reliability and consistency are highly valued

Timetable Design - Complexity and Challenge (2)

Airlines usually build their next season's scheduled based on previous one, and just adapting it (minor changes):

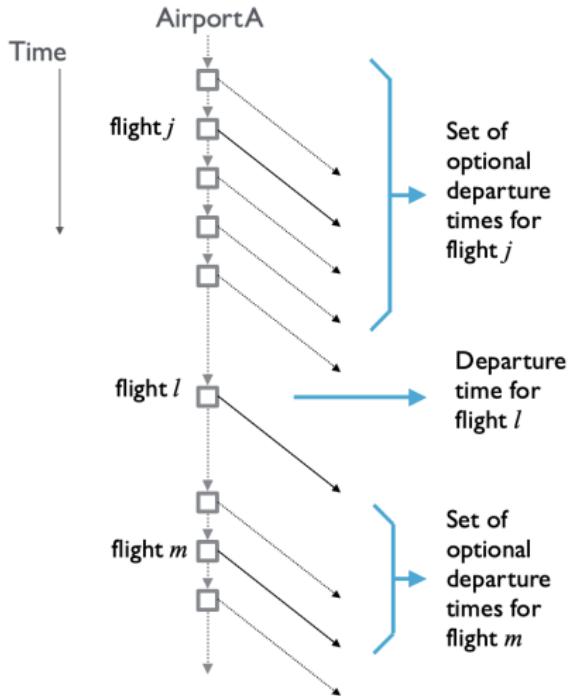
- Their historical booking data can be reused
- No changes in airport investments
- Airport slots can be maintained or easily renegotiated
- Consistency can be maintained or not dramatically altered

Scheduled design methods make use of FAM to decide:

- **Case 1:** Best departure times for a set of flights
- **Case 2:** Selection of flights to operate from an optional flight list

Timetable Design - Fleet Assignment Model Adaptations

Case 1: Best departure times for a set of flights



New notation

L^F : set of flights with fixed departure times
 $(L^F \ni \{l\})$

L^O : set of flights with optional departure times
 $(L^O \ni \{j, m\})$

L_i^O : set of optional departure times for flight i

Timetable Design - FAM Adaptations - Formulation

Case 1: Best departure times for a set of flights

Objective Function:

$$\min \sum_{i \in L} \sum_{k \in K} c_{k,i} \times f_i^k \quad (\text{OF})$$

Subject to:

$$\sum_{k \in K} f_i^k = 1, \quad \forall i \in L^F \quad (\text{C1.F})$$

$$\sum_{k \in K} \sum_{m \in L_i^O} f_m^k = 1, \quad \forall i \in L^O \quad (\text{C1.O})$$

$$y_{n^+}^k + \sum_{i \in O(k,n)} f_i^k - y_{n^-}^k - \sum_{i \in I(K,n)} f_i^k = 0, \quad \forall i \in N^k, k \in K \quad (\text{C2})$$

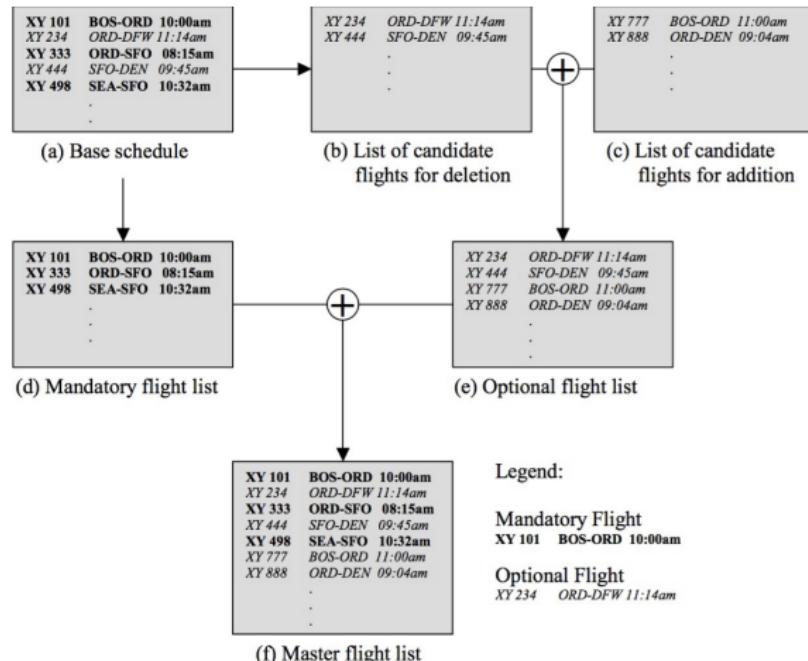
$$\sum_{a \in NG^k} y_a^k + f_a^k \leq AC^k, \quad \forall k \in K \quad (\text{C3})$$

$$f_i^k \in \{0, 1\}, \quad \forall k \in K, \forall i \in L$$

$$y_n^k \geq 0, \quad \forall n \in G^k, k \in K$$

Timetable Design - FAM Adaptations (2)

Case 2: Selection of flights to operate from a flight list



Proposed in: Lohatepanont, M. and Barnhart, C. (2004) 'Airline Schedule Planning: Integrated Models and Algorithms for Schedule Design and Fleet Assignment', Transportation Science 38(1):19-32.

Timetable Design - FAM Adaptations (3)

Case 2: Selection of flights to operate from a flight list

Sets:

P : set of itineraries in our portfolio, including the 'fictitious' itinerary

P_O : set of itineraries containing optional flight legs

L^F : set of mandatory flights

L_O : set of optional flights

$L(q)$: set of flight legs in itinerary q

Parameters:

N_q : the number of flights in itinerary q

Decision Variables:

Z_q : 1 if itinerary q is included in the flight timetable, and 0 otherwise

Itinerary-based Fleet Assignment Problem - Model

Objective Function:

$$\min \sum_{i \in L} \sum_{k \in K} c_{k,i} \times f_i^k + \sum_{p \in P} \sum_{r \in P} (fare_p - b_p^r \times fare_r) \times t_p^r + \sum_{q \in P^O} fare_p \times D_q + \times (1 - Z_q) \quad (\text{OF})$$

Subject to:

$$\sum_{k \in K} f_i^k = 1, \quad \forall i \in L^F \quad (\text{C1.F})$$

$$\sum_{k \in K} f_i^k \leq 1, \quad \forall i \in L^O \quad (\text{C1.O})$$

$$y_{n+}^k + \sum_{i \in O(k,n)} f_i^k - y_n^k - \sum_{i \in I(k,n)} f_i^k = 0, \quad \forall n \in N^k, k \in K \quad (\text{C2})$$

$$\sum_{a \in NG^k} y_a^k + f_a^k \leq AC^k, \quad \forall k \in K \quad (\text{C3})$$

$$\sum_{k \in K} s_k \times f_i^k + \sum_{p \in P} \sum_{r \in P} \delta_p^r \times t_p^r - \sum_{r \in P} \sum_{p \in P} \delta_i^p \times b_r^p \times t_r^p \geq Q_i, \quad \forall i \in L \quad (\text{C4})$$

$$\sum_{r \in P} t_p^r \leq D_p, \quad \forall p \in P \quad (\text{C5})$$

$$Z_q - \sum_{k \in K} f_i^k \leq 0, \quad \forall i \in L(q) \quad (\text{C6})$$

$$Z_q - \sum_{i \in L(q)} \sum_{k \in K} f_i^k \geq 1 - N_q, \quad \forall q \in P^O \quad (\text{C7})$$

$$f_i^k \in \{0, 1\}, \forall k \in K \quad y_n^k \geq 0, \forall n \in G^k, k \in K \quad Z_q \in \{0, 1\}, \forall q \in P^O \quad t_p^r \geq 0, \forall p, r \in P$$

Airline Planning and Optimisation - AE4423-20

Lecture 6

Aircraft Routing & Dynamic Programming

Marta Ribeiro
Assistant Professor
Air Transport & Operation
Control & Operations

Delft University of Technology, The Netherlands

November, 2024

Course Information - Content

Management

- Planning structure
- Performance Indicators
- Demand and Supply
- Market share

Tactical Planning

- Passenger Mix Flow
- Fleet Assignment
- Aircraft Rotation**
- Crew Scheduling
- Maintenance



Strategical Planing

- Network Structure
- Demand Forecast
- Network Planning
- Fleet Planning

Operations Research

MILP Models

- Multi-commodity Flow Problems
- Shortest-path Algorithms
- Column Generation Algorithm

Dynamic Programming

Outline - Lecture 6

① Aircraft Routing

- ② a Maintenance Routing
- ② b Example
- ② c Approach
- ② d Results

② Aircraft Routing + Timetable

- ③ a Dynamic Programming
- ③ b Example
- ③ c Approach
- ③ d Results
- ③ e Dynamic Programming - Extra Information

Problem Definition

Fleet assignment determines the **aircraft type** to operate on a specific flight.



Aircraft routing determines which **specific aircraft in the fleet** operates each flight in our timetable. The main objectives of aircraft routing are:

- To cover each flight by only one aircraft
- To balance the aircraft utilisation
- To comply with maintenance requirements

Problem Definition (2)

Different possible objective functions:

- Minimize the total operating cost for a specific aircraft type
- Maximize maintenance opportunities

Commonly subject to the following considerations:

- Flight coverage (operate each flight once and only once)
- Route continuity – space and time
- Number of available aircraft
- Maintenance and airworthiness requirements
- Minimum turn-around time at the airports
- Aircraft utilisation balance for uniform fleet wear and tear

Maintenance Routing

Also called 'Aircraft maintenance routing problem':

- Aircraft undergo periodic maintenance checks at regular intervals
- These checks are performed at specific maintenance stations
- In general, regulations require inspections of aircraft approximately every **3–5 days**

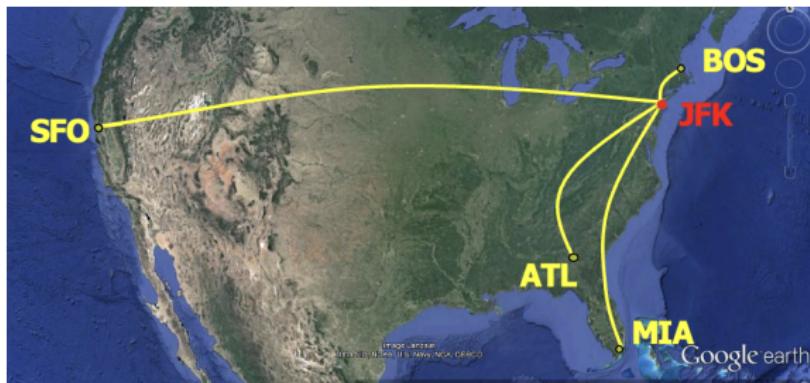
Airlines typically built aircraft routing that provide 'frequent' maintenance opportunities at the maintenance station:

- Provide slack times at the station
- Guarantee enough visits to the station every 3-5 days

Maintenance Routing - Example

Define the route of 6 aircraft operating flights for an airline with the maintenance base at JFK:

Flight	From	Dep.	To	Arrival	Hrs
125	JFK	7:25	SFO	9:55	5.5
110	ATL	8:10	JFK	10:40	2.5
113	MIA	9:10	JFK	12:10	3
131	JFK	9:30	ATL	12:00	2.5
105	SFO	10:50	JFK	18:20	5.5
138	JFK	12:30	BOS	14:00	2.5
111	ATL	13:10	JFK	17:30	3
114	MIA	14:30	JFK	17:30	3
118	BOS	15:00	JFK	16:30	1.5
135	JFK	15:10	MIA	18:10	3
133	JFK	18:05	ATL	20:35	2.5
136	JFK	18:10	MIA	21:10	3



Based on: Airline Operations and Scheduling, Bazargan, M. (2010)

Maintenance Routing - Example (1)

Creation of routes (lines of flights):

- Three day closed time-and-space sequence of flights
- Consideration of turn-around-times of 45 minutes at each airport

No feasible continuation	Flight no.	From	Dep Time	To	Arrival Time	Hrs
	113	MIA	09:10	JFK	12:10	3
	138	JFK	12:30	BOS	14:00	2.5
	135	JFK	15:10	MIA	18:10	3
	133	JFK	18:05	ATL	20:35	2.5
	136	JFK	18:10	MIA	21:10	3

Maintenance Routing - Example - Approach

Creation of routes (lines of flights):

- Each aircraft is routed such that it stays overnight at JFK at least once every three days
- Valid route starts and ends at the same airport, and includes at least one overnight at JFK for maintenance
- Example:

Flight no.	From	Dep Time	To	Arrival Time	Hrs
Day 1					
125	JFK	07:25	SFO	09:55	5.5
Day 2					
105	SFO	10:50	JFK	18:20	5.5
Day 3					
131	JFK	09:30	ATL	12:00	2.5
111	ATL	13:10	JFK	17:30	3

Maintenance Routing - Example - Approach (2)

Flight coverage consideration:

- Each flight per day should be operated at least once and only once

Flight no.	From	To	Departure Time (EST)	Arrival Time (EST)	Hours	Day
125	JFK	7:25	SFO	9:55	5.5	1
125	JFK	17:25	SFO	9:55	5.5	2
125	JFK	17:25	SFO	9:55	5.5	3

- Only six routing candidates (of the 455 possible) cover flight 125 in one of the days (and visit JFK, at least once):

Candidate	Day 1	Day 2	Day 3
x_1	JFK-SFO	SFO-JFK	JFK-ATL-JFK
x_2	JFK-SFO	SFO-JFK	JFK-BOS-JFK
x_3	JFK-ATL-JFK	JFK-SFO	SFO-JFK
x_4	JFK-BOS-JFK	JFK-SFO	SFO-JFK
x_5	SFO-JFK	JFK-ATL-JFK	JFK-SFO
x_6	SFO-JFK	JFK-BOS-JFK	JFK-SFO

Maintenance Routing - Example - Approach (3)

Flight coverage consideration:

	Candidate	Day 1	Day 2	Day 3
$x_1 + x_2 = 1$	x_1	125	105	131-111
	x_2	125	105	138-118
$x_3 + x_4 = 1$	x_3	131-111	125	105
	x_4	138-118	125	105
$x_5 + x_6 = 1$	x_5	105	131-111	125
	x_6	105	138-118	125

Introduce $a_{i,j} = 1$ if flight i is covered by route j , and 0 otherwise:

$$\sum_{j=1}^R a_{i,j} \times x_j = 1, \quad \forall i \in F$$

Each flight will be covered by one, and only one, route.

Maintenance Routing - Approach

Sets:

R : Number of possible routings

F : Set of flights

Parameters:

m_j : Number of maintenance opportunities for route j

$a_{i,j}$: 1 if flight i is covered by route j , 0 otherwise

N : Total number of aircraft in fleet

Decision Variables:

x_j : 1 if route j is selected, 0 otherwise

Maintenance Routing - Model

Objective Function:

$$\max \sum_{j=1}^R m_j \times x_j$$

(OF) Maximize maintenance opportunities

Subject to:

$$\sum_{j=1}^R a_{i,j} \times x_j = 1, \quad \forall i \in F$$

(C1) Select routes covering flights once

$$\sum_{j=1}^R x_j \leq N$$

(C2) Do not select more routes than aircraft

$$x_j \in \{0, 1\}, \quad \forall j \in R$$

Model Size ($|F|=|\text{Flights}|$; $|R|=|\text{Routes}|$):

- Constraints: $|F| + 1$
- Variables: $|R|$

Maintenance Routing - Results

One of the possible alternative optimal solutions for 6 routes (airline has 6 aircraft):

Route	Day 1	Day 2	Day 3
1	125-105 JFK-SFO-JFK	135 JFK-MIA	114 MIA-JFK
2	110-138-118-136 ATL-JFK-BOS-JFK-MIA	113 MIA-JFK	131-111-133 JFK-ATL-JFK-ATL
3	113 MIA-JFK	131-111-133 JFK-ATL-JFK-ATL	110-138-118-136 ATL-JFK-BOS-JFK-MIA
4	131-111-133 JFK-ATL-JFK-ATL	110-138-118-136 ATL-JFK-BOS-JFK-MIA	113 MIA-JFK
5	114 MIA-JFK	125-105 JFK-SFO-JFK	135 JFK-MIA
6	135 JFK-MIA	114 MIA-JFK	125-105 JFK-SFO-JFK

Objective function value = 9

Legend: Maintenance Opportunities

Maintenance Routing - Results (2)

- Very simple model – where is the complexity?
 - All routes must be generated before solving the aircraft routing problem (computationally demanding)
- Balanced utilisation per route selected is not considered
 - The result is (by chance) balanced - 17-19 hours per route (3 days)
- Aircraft tailor numbers still have to be associated with each route
 - A rotation approach should force aircraft to have similar rotations and ageing
 - Each aircraft follows a 18-days sequence of routes:

	Days 1-3	Days 4-6	Days 7-9	Days 10-12	Days 13-15	Days 16-18
AC1	Route 3	Route 1	Route 4	Route 6	Route 2	Route 5
AC2	Route 1	Route 4	Route 6	Route 2	Route 5	Route 3
AC3	Route 4	Route 6	Route 2	Route 5	Route 3	Route 1
AC4	Route 6	Route 2	Route 5	Route 3	Route 1	Route 4
AC5	Route 2	Route 5	Route 3	Route 1	Route 4	Route 6
AC6	Route 5	Route 3	Route 1	Route 4	Route 6	Route 2

Outline - Lecture 6

① Aircraft Routing

- ② a Maintenance Routing
- ② b Example
- ② c Approach
- ② d Results

② Aircraft Routing + Timetable

- ② a Dynamic Programming
- ② b Example
- ② c Approach
- ② d Results
- ② e Dynamic Programming - Extra Information

Aircraft Routing + Timetable

The complexity of the set-partitioning approach is the generation of routes. The generation of all possible routes is very time demanding.

However, the routing problem is a dynamic network problem:

- **Dynamic** – route over time
- **Network** – route over space

An alternative solution method is **Dynamic Programming**:

- Developed by Richard Bellman in 1950s
- Used in the Reinforcement Learning algorithm (machine learning), also called Approximate Dynamic Programming algorithm

More Info

Dynamic Programming

Dynamic Programming is a useful mathematical technique for making a sequence of interrelated decisions.

Divide-and-conquer concept:

- The idea: to break a large problem down into incremental steps recursively solved to obtain the solution for the large problem.
- These steps are sub-problems that have a 'trivial' solution (i.e., simple to computer)

Property required - Markov decision process:

- The environment is fully observable
- Future decisions and rewards only depend on the present and it is not influenced by past decisions

Dynamic Programming (2)

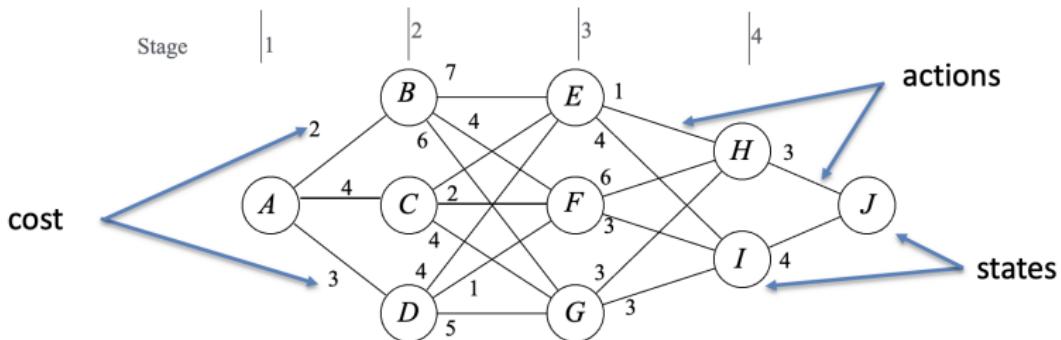
Bellman's Principle of Optimality

An **optimal policy** has the property that whatever the initial **state** and initial **decision** are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision ^a.

^aR. E. Bellman, Chap. III.3, Dynamic Programming. Princeton University Press, Princeton, NJ, 1957

- **Policy:** decision rule defined according to the current state
- **State:** current situation of the system, including all the information needed to characterise it
- **Decision (or action):** action from the decision maker to influence the evolution of the system

Dynamic Programming (3)



$f(s_n, x_n)$ is the total cost of the best overall policy for the remaining stages:

$$f(s_n, x_n) = \underbrace{c(s_n, x_n)}_{\text{immediate cost (stage } n\text{)}} + \underbrace{\min_{x_{n+1}} f^*(s_{n+1}, x_{n+1})}_{\text{min future cost (stage } n+1 \text{ onward)}}$$

Where:

- s is the state and s_n is a state at stage n
- n is the stage (stage 1: A; stage 2: B, C, D; ...)
- x_n is the action taken at stage n (i.e., next move)
- $c(s_n, x_n)$ is the immediate contribution (cost) of the action taken

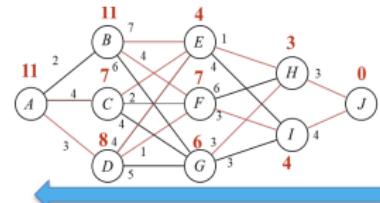
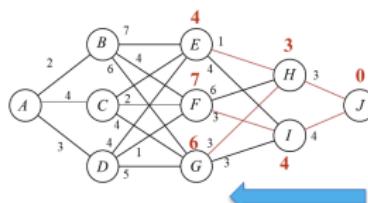
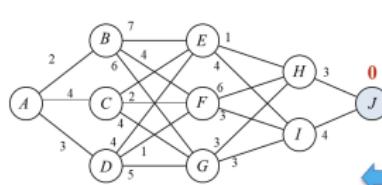
Source: Introduction to Operations Research – Hillier & Lieberman (Ch.11)

Dynamic Programming (4)

How to compute the optimal solution for any s ?

$$f(s_n, x_n) = \underbrace{c(s_n, x_n)}_{\text{Cost of current state}} + \underbrace{\min[c(s_{n+1}, x_{n+1}) + c(s_{n+2}, x_{n+2}) + c(s_{n+3}, x_{n+3}) + \dots]}_{\text{Cost of future state}}$$

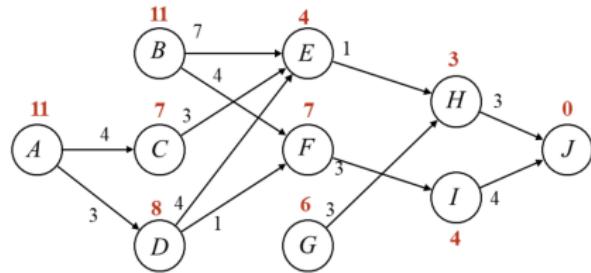
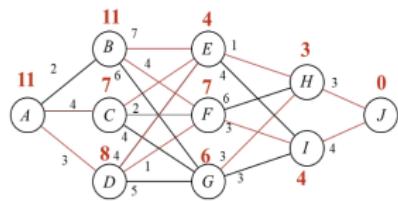
- ① Start with an end node ('J'), for which $c(s_J, x_J)$ is known (normally, $c(J, J) = 0$)
- ② Progress backwards computing minimum costs, all the way to the first node



Dynamic Programming (5)

How to compute the optimal solution for any s ?

- ① Start with an end node ('J'), for which $c(s_J, x_J)$ is known
- ② Progress backwards computing minimum costs >> Map of cost for the full network of states and actions
- ③ Compute the minimum path between the start node and the end node using a **greedy policy** (optimal policy)



Example for node A:

$$f_1(A, \rightarrow B) = 2 + \min_{n+1} f^*(B, \rightarrow E \text{ or } \rightarrow F \text{ or } \rightarrow G) = 2 + 11 = 13$$

$$f_1(A, \rightarrow C) = 4 + \min_{n+1} f^*(C, \rightarrow E \text{ or } \rightarrow F \text{ or } \rightarrow G) = 4 + 7 = 11$$

$$f_1(A, \rightarrow D) = 3 + \min_{n+1} f^*(D, \rightarrow E \text{ or } \rightarrow F \text{ or } \rightarrow G) = 3 + 8 = 11$$

Optimal

Dynamic Programming - Example

An airline is operating between six airports in Europe. Hub airport is in Amsterdam



	Demand per Day						
	EHAM	EDDF	LEMD	LIRF	EIDW	ESSA	LPPT
EHAM	0	582	168	204	288	204	9
EDDF	582	0	180	303	210	132	84
LEMD	168	180	0	357	165	105	348
LIRF	204	303	357	0	195	132	66
EIDW	288	210	165	195	0	171	72
ESSA	204	132	105	132	171	0	3
LPPT	9	84	348	66	72	3	0

Variation per hour ($Dem(t)_{i,j} = Dem_{i,j} * Coef(t)_i$)

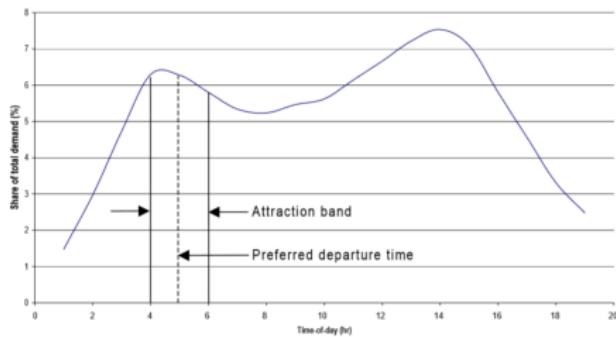
Airport	Hour																							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
EHAM	0	0	0	0	0.0943	0.0991	0.1537	0.2141	0.0756	0.0910	0.2018	0.1194	0.2179	0.0825	0.1186	0.1116	0.3119	0.2255	0.1575	0.1299	0.0886	0.0519	0.1187	0.0377
EDDF	0	0	0	0	0.0394	0.0387	0.0973	0.2456	0.4103	0.1999	0.0837	0.0840	0.1125	0.0974	0.1331	0.1314	0.1907	0.1767	0.2229	0.0678	0.2482	0.1083	0.0703	0.0219
LEMD	0	0	0	0	0.0137	0.0233	0.0344	0.0588	0.1531	0.2205	0.4041	0.1657	0.1747	0.0819	0.0548	0.0782	0.1140	0.2163	0.1370	0.0502	0.0252	0.0000	0.0000	0.0000
LIRF	0	0	0	0	0.0703	0.0830	0.1210	0.0939	0.1120	0.0933	0.1460	0.1003	0.0937	0.1032	0.1364	0.1510	0.4576	0.1950	0.1841	0.1561	0.0639	0.0861	0.0000	0.0000
EIDW	0	0	0	0	0.0844	0.0594	0.1022	0.2126	0.1777	0.0865	0.0542	0.1022	0.0780	0.0326	0.0574	0.0659	0.4562	0.1925	0.2895	0.1267	0.0805	0.0771	0.0000	0.0000
ESSA	0	0	0	0	0.1158	0.1316	0.1431	0.3203	0.2942	0.0940	0.0531	0.0894	0.0816	0.0562	0.1032	0.1096	0.3006	0.1872	0.2963	0.1600	0.0454	0.0968	0.0751	0.0000
LPPT	0	0	0	0	0.0946	0.0751	0.1599	0.0892	0.1175	0.1505	0.2275	0.0753	0.1633	0.0537	0.1357	0.3080	0.2637	0.0966	0.2802	0.0471	0.0681	0.0836	0.0222	0.0000

Dynamic Programming - Example (2)

Fleet:

	Aircraft Type	Type 2	Type 3
Speed [km/h]	810	870	
Seats	50	120	
Average TAT [min]	30	35	
Maximum Range [km]	3800	4500	
Runway Required [m]	1500	1600	
Lease Cost [€]	4540	7340	
Fixed Operating Cost (Per Flight Leg) [€]	620	920	
Cost per Hour	710	1000	
Fuel Cost Parameter	2.4	2.9	
Fleet	3	2	

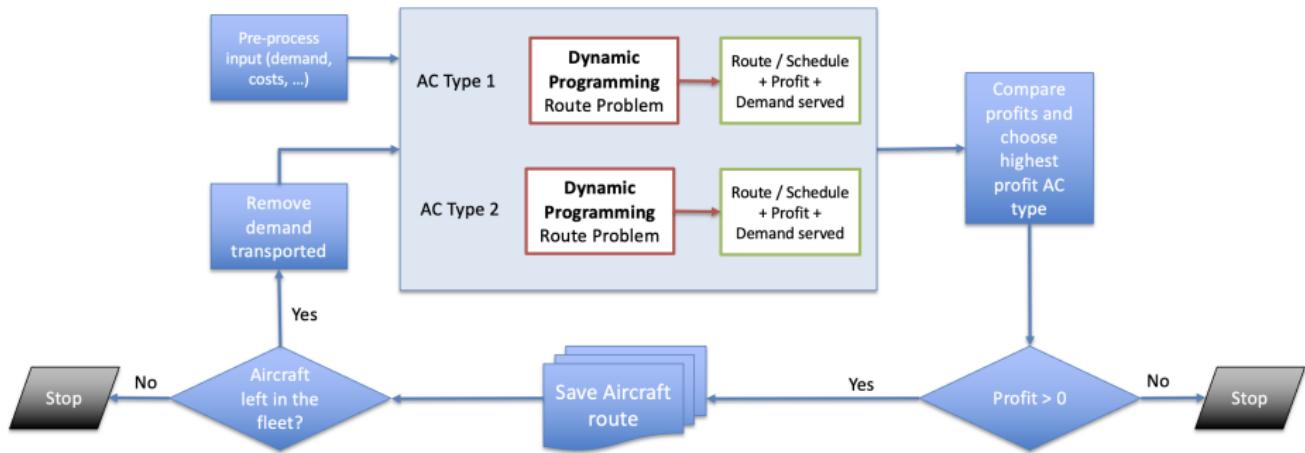
Recapture time band:



Determine the best timetable and aircraft route, assuming that:

- Aircraft start and end their daily routes at the hub
- Only hub-spoke routes are considered
- No connections - demand is given per flight route
- Passengers can adapt their flight time by max 2 hours (recapture time band)
- Assume time segments of 6 min (discretization of time)

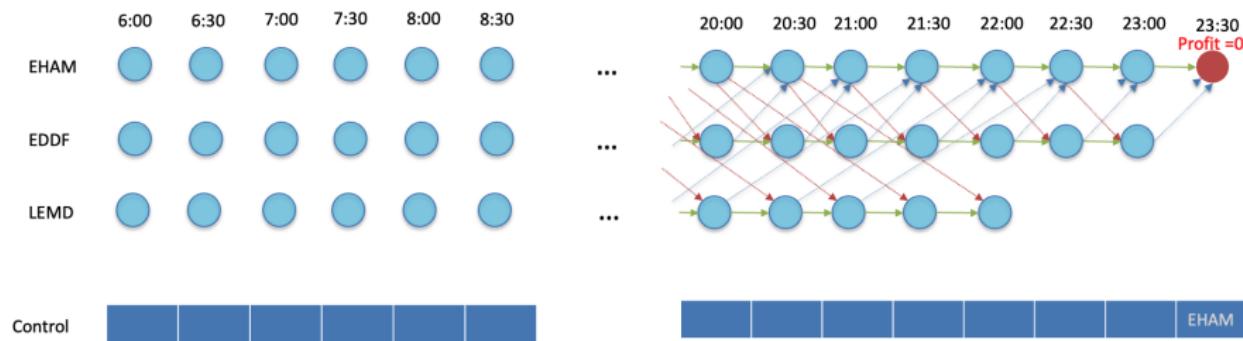
Dynamic Programming - Approach



Dynamic Programming - Approach (2)

Dynamic Programming Approach (time discretisation 30 min and only 3 airports)

- Ground arcs
- Flight arcs to the hub
- Flight arcs from the hub

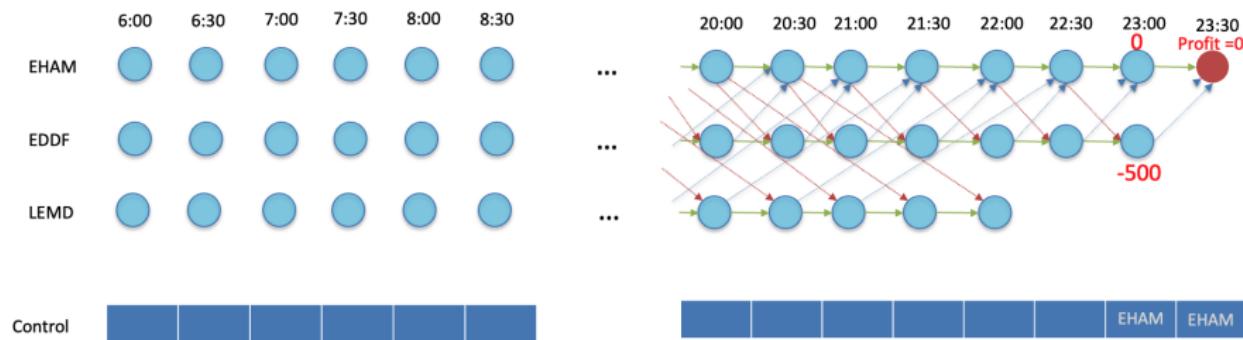


We have to start at the hub (EHAM) so the total profit will be 2870 units of profit

Dynamic Programming - Approach (3)

Dynamic Programming Approach (time discretisation 30 min and only 3 airports)

- Ground arcs
- Flight arcs to the hub
- Flight arcs from the hub

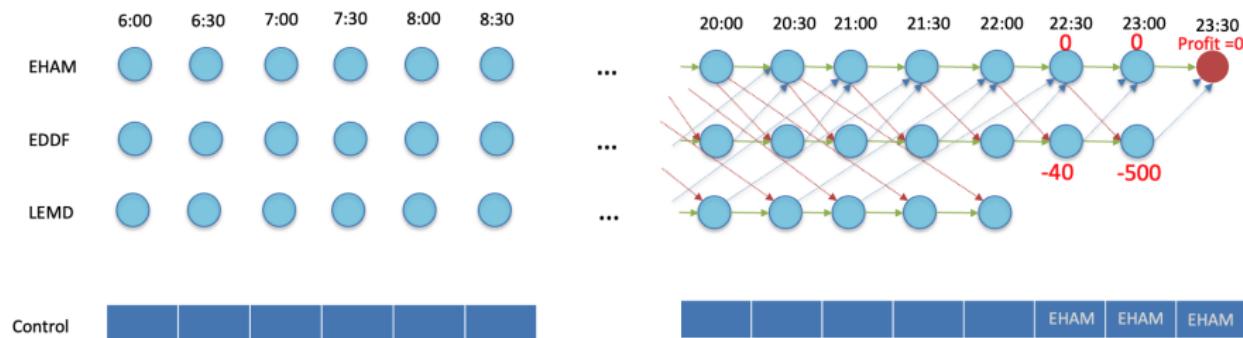


We have to start at the hub (EHAM) so the total profit will be 2870 units of profit

Dynamic Programming - Approach (4)

Dynamic Programming Approach (time discretisation 30 min and only 3 airports)

- Ground arcs
- Flight arcs to the hub
- Flight arcs from the hub

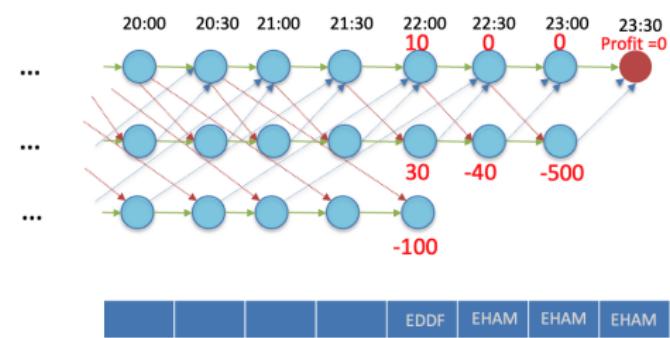
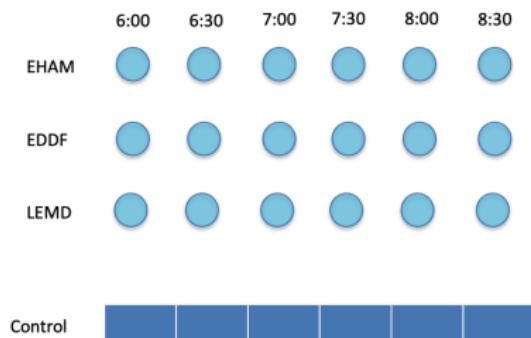


We have to start at the hub (EHAM) so the total profit will be 2870 units of profit

Dynamic Programming - Approach (5)

Dynamic Programming Approach (time discretisation 30 min and only 3 airports)

- Ground arcs
- Flight arcs to the hub
- Flight arcs from the hub

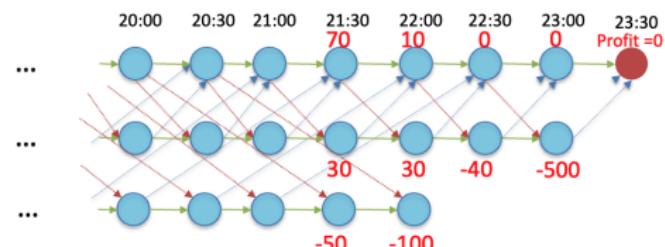
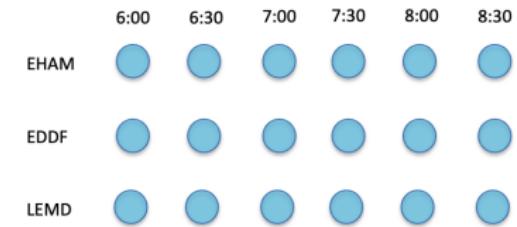


We have to start at the hub (EHAM) so the total profit will be 2870 units of profit

Dynamic Programming - Approach (6)

Dynamic Programming Approach (time discretisation 30 min and only 3 airports)

- Ground arcs
- Flight arcs to the hub
- Flight arcs from the hub

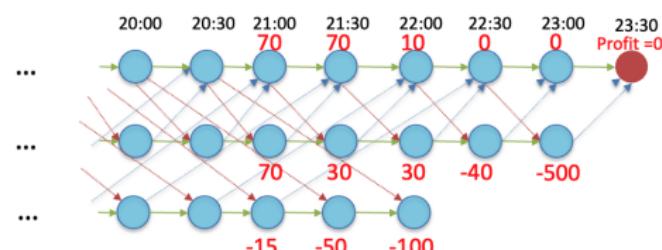
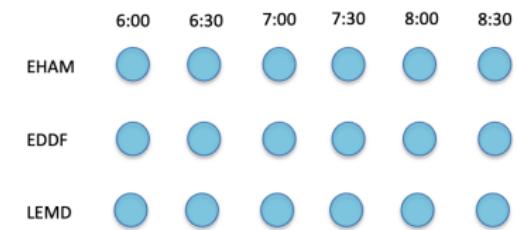


We have to start at the hub (EHAM) so the total profit will be 2870 units of profit

Dynamic Programming - Approach (7)

Dynamic Programming Approach (time discretisation 30 min and only 3 airports)

- Ground arcs
- Flight arcs to the hub
- Flight arcs from the hub

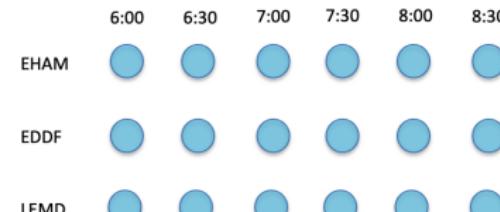


We have to start at the hub (EHAM) so the total profit will be 2870 units of profit

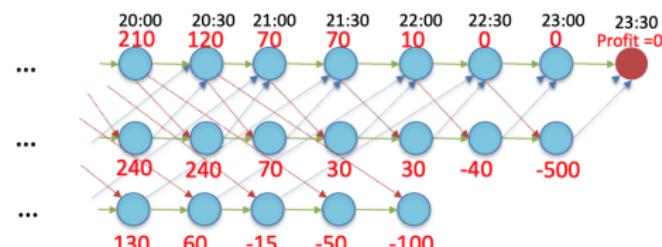
Dynamic Programming - Approach (8)

Dynamic Programming Approach (time discretisation 30 min and only 3 airports)

- Ground arcs
- Flight arcs to the hub
- Flight arcs from the hub



Control



LEMD EDDF EHAM EDDF EDDF EHAM EHAM EHAM

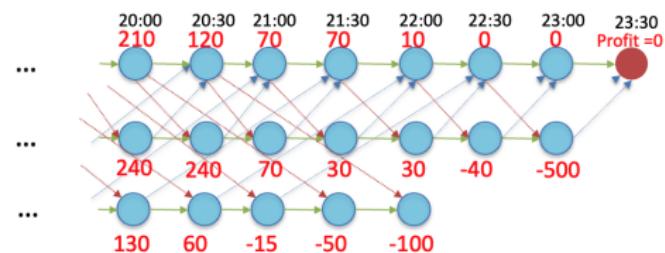
We have to start at the hub (EHAM) so the total profit will be 2870 units of profit

Dynamic Programming - Approach (9)

Dynamic Programming Approach (time discretisation 30 min and only 3 airports)

- Ground arcs
- Flight arcs to the hub
- Flight arcs from the hub

	6:00	6:30	7:00	7:30	8:00	8:30
EHAM						
	2870	2460	2390	2110	1970	1970
EDDF						
	2640	2530	2190	1850	1810	1730
LEMD						
	2390	2140	2040	1900	1900	1780



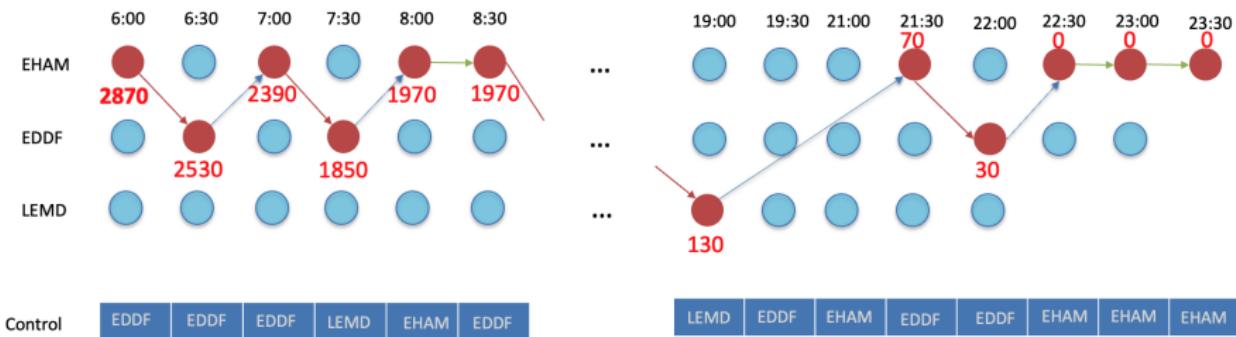
Control EDDF EDDF EDDF LEMD EHAM EDDF

LEMD EDDF EHAM EDDF EDDF EHAM EHAM EHAM

We have to start at the hub (EHAM) so the total profit will be 2870 units of profit

Dynamic Programming - Approach (10)

Dynamic Programming Approach (time discretisation 30 min and only 3 airports)



Final Route / Schedule according to the greedy policy (i.e., the high profit path)

Dynamic Programming - Results

- Total Profit – 29683 €
- All aircraft in the fleet are used Routes for the first 3 aircraft:

Aircraft	Type	Total profit Utilization [h/day]		
Aircraft 1:	Type 2	14268	9.21	

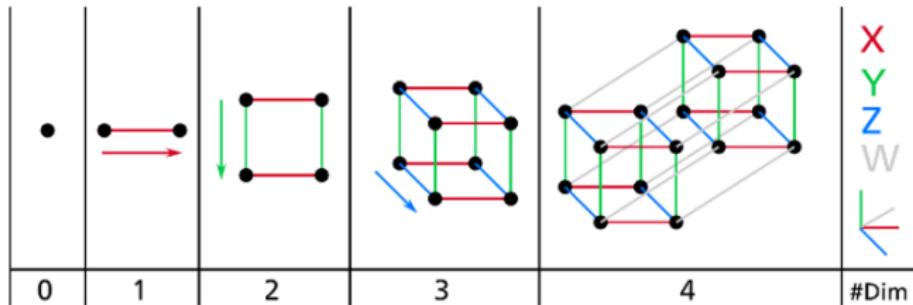
Departure Time	Route	Arrival Tim	Passenger	Load-factor	Profit [\$]
6 h 18	EHAM - EDDF	7 h 12	120	1	2064.47
7 h 42	EDDF - EHAM	8 h 36	120	1	2064.47
9 h 06	EHAM - EDDF	10 h 00	118	0.98	1984.27
10 h 30	EDDF - EHAM	11 h 24	49	0.41	-782.56
11 h 54	EHAM - EDDF	12 h 48	120	1	2064.47
13 h 54	EDDF - EHAM	14 h 48	78	0.65	380.31
15 h 42	EHAM - EDDF	16 h 36	120	1	2064.47
17 h 06	EDDF - EHAM	18 h 00	120	1	2064.47
18 h 30	EHAM - EDDF	19 h 24	76	0.63	300.11
19 h 54	EDDF - EHAM	20 h 48	120	1	2064.47

Aircraft	Type	Total profit Utilization [h/day]		
Aircraft 2:	Type 2	11863	6.41	
Departure Time	Route	Arrival Tim	Passenger	Load-factor
6 h 30	EHAM - EDDF	7 h 24	120	1
7 h 54	EDDF - EHAM	8 h 48	120	1
11 h 54	EHAM - EIDW	13 h 12	63	0.53
15 h 06	EIDW - EHAM	16 h 24	120	1
16 h 54	EHAM - EDDF	17 h 48	120	1
19 h 54	EDDF - EHAM	20 h 48	120	1

Aircraft	Type	Total profit Utilization [h/day]		
Aircraft 3:	Type 1	1412	8.6	

Departure Time	Route	Arrival Tim	Passenger	Load-factor	Profit [\$]
6 h 00	EHAM - EIDW	7 h 24	50	1	259.48
7 h 54	EIDW - EHAM	9 h 18	50	1	259.48
11 h 54	EHAM - EDDF	12 h 48	50	1	52.02
14 h 06	EDDF - EHAM	15 h 00	50	1	52.02
15 h 30	EHAM - ESSA	17 h 24	50	1	394.52
17 h 54	ESSA - EHAM	19 h 48	50	1	394.52

Dynamic Programming - Curse of Dimensionality



Term defined by R. Bellman in his book "Dynamic Programming" (1957):

- The state space is too large
- The action space is too large
- Computing the expected 'future' costs is impracticable

(Some) Solutions:

- Feature selection
- Approximate Dynamic Programming (ADP): Global optimum is not guaranteed

AE4423-20: *Airline Planning and Optimization - the Column Generation algorithm*

Alessandro Bombelli
a.bombelli@tudelft.nl



Delft University of Technology, The Netherlands

December 5, 2024



Underlying intuition

Let us consider a **Capacitated Vehicle Routing Problem with Time Windows (CVRP-TW)** with a depot $\mathcal{D} = \{0\}$ and 7 customers $c \in \mathcal{C} = \{1, \dots, 7\}$, so that the full set of nodes is $\mathcal{N} = \mathcal{D} \cup \mathcal{C} = \{0, \dots, 7\}$. In addition, there are 3 available identical vehicles $k \in \mathcal{K} = \{1, 2, 3\}$

Underlying intuition

Let us consider a **Capacitated Vehicle Routing Problem with Time Windows (CVRP-TW)** with a depot $\mathcal{D} = \{0\}$ and 7 customers $c \in \mathcal{C} = \{1, \dots, 7\}$, so that the full set of nodes is $\mathcal{N} = \mathcal{D} \cup \mathcal{C} = \{0, \dots, 7\}$. In addition, there are 3 available identical vehicles $k \in \mathcal{K} = \{1, 2, 3\}$

Let us also assume each customer has a **demand request** equal to D_c , while D_{ij} represents the distance between any $(i, j) \in \mathcal{N} \times \mathcal{N}$. Each vehicle has a **maximum transportable capacity** equal to Q_k and a **maximum range** equal to R_k (for the sake of)

Underlying intuition

Let us consider a **Capacitated Vehicle Routing Problem with Time Windows (CVRP-TW)** with a depot $\mathcal{D} = \{0\}$ and 7 customers $c \in \mathcal{C} = \{1, \dots, 7\}$, so that the full set of nodes is $\mathcal{N} = \mathcal{D} \cup \mathcal{C} = \{0, \dots, 7\}$. In addition, there are 3 available identical vehicles $k \in \mathcal{K} = \{1, 2, 3\}$

Let us also assume each customer has a **demand request** equal to D_c , while D_{ij} represents the distance between any $(i, j) \in \mathcal{N} \times \mathcal{N}$. Each vehicle has a **maximum transportable capacity** equal to Q_k and a **maximum range** equal to R_k (for the sake of)

Our goal is to determine **how to deploy our fleet so that every customer is served while, for example, minimizing the traveled distance**

Underlying intuition

Let us consider a **Capacitated Vehicle Routing Problem with Time Windows (CVRP-TW)** with a depot $\mathcal{D} = \{0\}$ and 7 customers $c \in \mathcal{C} = \{1, \dots, 7\}$, so that the full set of nodes is $\mathcal{N} = \mathcal{D} \cup \mathcal{C} = \{0, \dots, 7\}$. In addition, there are 3 available identical vehicles $k \in \mathcal{K} = \{1, 2, 3\}$

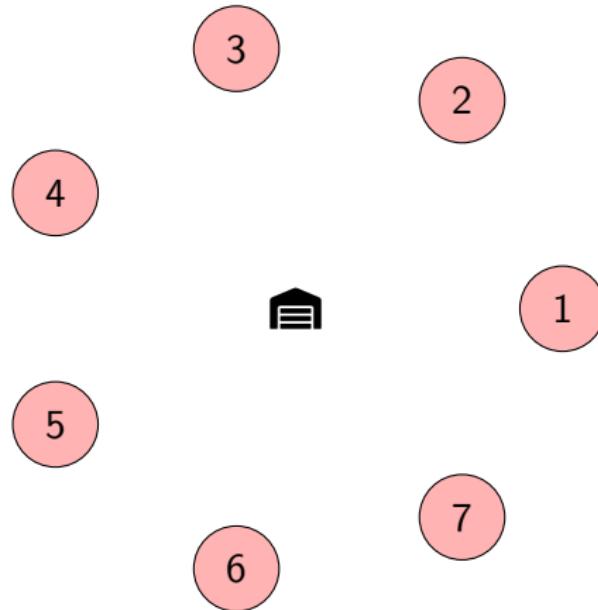
Let us also assume each customer has a **demand request** equal to D_c , while D_{ij} represents the distance between any $(i, j) \in \mathcal{N} \times \mathcal{N}$. Each vehicle has a **maximum transportable capacity** equal to Q_k and a **maximum range** equal to R_k (for the sake of)

Our goal is to determine **how to deploy our fleet so that every customer is served while, for example, minimizing the traveled distance**

We also have a time-window $[E_c, L_c]$ when each customer $c \in \mathcal{C}$ can be visited. hence, we need **routing binary decision variables** $x_{ij}^k \forall (i, j) \in \mathcal{N} \times \mathcal{N}, k \in \mathcal{K}$ **and time-related continuous decision variables** $t_c \forall c \in \mathcal{C}$

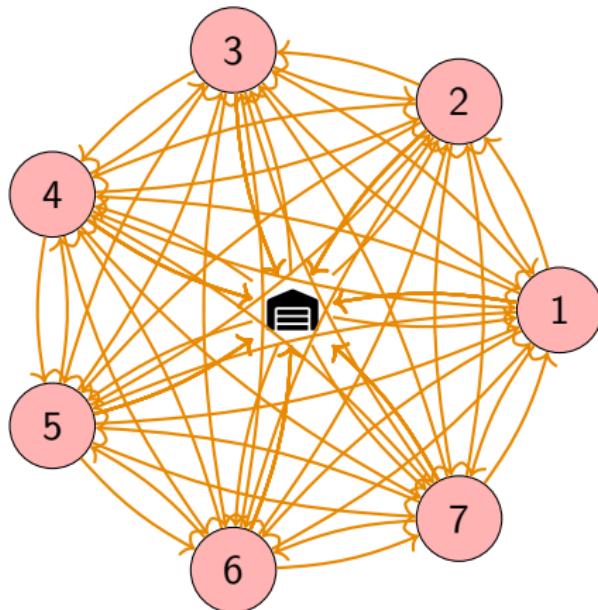
Underlying intuition

Assuming we can reach every node from every other node, we can define our problem on a complete graph going from here ...



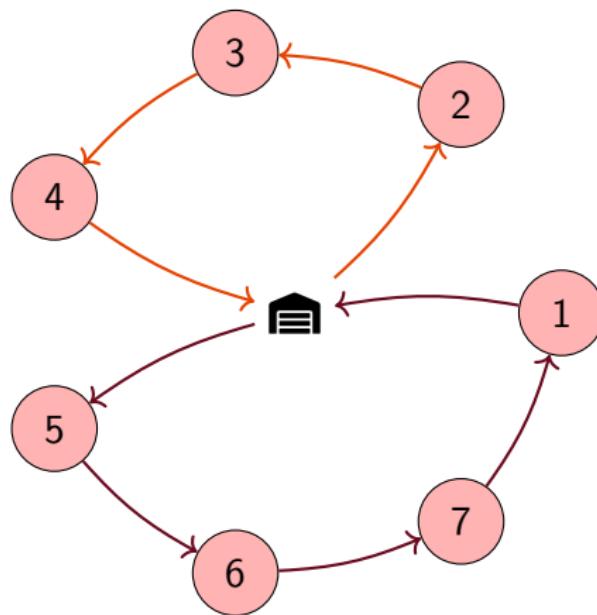
Underlying intuition

... to here, where 168 decision variables are defined (56 arcs, each potentially used by each of the 3 vehicles): $|\mathcal{N}| \times |\mathcal{N}-1| \times |\mathcal{K}| = 8 \times 7 \times 3 = 168$



Underlying intuition

Let us assume that solving this problem yields the following optimal solution



Underlying intuition

For this specific instance/example, we only need 2 vehicles out of 3 (let us assume vehicles 1 and 2) and only 9 routing decision variables are needed.

Underlying intuition

For this specific instance/example, we only need 2 vehicles out of 3 (let us assume vehicles 1 and 2) and only 9 routing decision variables are needed. Hence, $\frac{159}{168} \times 100 \simeq 95\%$ of the defined decision variables were useless in the end!

Underlying intuition

For this specific instance/example, we only need 2 vehicles out of 3 (let us assume vehicles 1 and 2) and only 9 routing decision variables are needed. Hence, $\frac{159}{168} \times 100 \simeq 95\%$ of the defined decision variables were useless in the end!

Let us assume we re-wrote our CVRP-TW using, as decision variables, whether a vehicle follows a pre-defined path instead of a single arc.

Underlying intuition

For this specific instance/example, we only need 2 vehicles out of 3 (let us assume vehicles 1 and 2) and only 9 routing decision variables are needed. Hence, $\frac{159}{168} \times 100 \simeq 95\%$ of the defined decision variables were useless in the end!

Let us assume we re-wrote our CVRP-TW using, as decision variables, whether a vehicle follows a pre-defined path instead of a single arc. For example, let us consider we have computed the following paths for each of the vehicles (note: \mathcal{P}_k defines the set of paths for vehicle k):

Underlying intuition

For this specific instance/example, we only need 2 vehicles out of 3 (let us assume vehicles 1 and 2) and only 9 routing decision variables are needed. Hence, $\frac{159}{168} \times 100 \simeq 95\%$ of the defined decision variables were useless in the end!

Let us assume we re-wrote our CVRP-TW using, as decision variables, whether a vehicle follows a pre-defined path instead of a single arc. For example, let us consider we have computed the following paths for each of the vehicles (note: \mathcal{P}_k defines the set of paths for vehicle k):

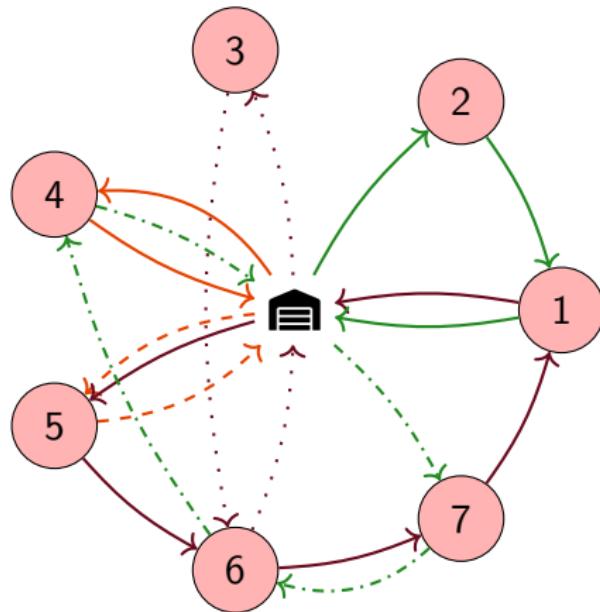
① **vehicle 1:** $\mathcal{P}_1 = \underbrace{\{(0, 4, 0), (0, 5, 0)\}}_1$

② **vehicle 2:** $\mathcal{P}_2 = \underbrace{\{(0, 5, 6, 7, 1, 0)\}}_1, \underbrace{\{(0, 3, 6, 0)\}}_2$

③ **vehicle 3:** $\mathcal{P}_3 = \underbrace{\{(0, 2, 1, 0)\}}_1, \underbrace{\{(0, 7, 6, 4, 0)\}}_2$

Underlying intuition

Which results in this set of routes (with the same color we identify all the potential routes for a given vehicle)



Underlying intuition

We could define a new set of decision variables $y_{k,p} \in \{0, 1\}$, where $y_{k,p}$ is unitary if vehicle k follows path p .

Underlying intuition

We could define a new set of decision variables $y_{k,p} \in \{0, 1\}$, where $y_{k,p}$ is unitary if vehicle k follows path p . With such a setting and given our definition of \mathcal{P}_1 , \mathcal{P}_2 , and \mathcal{P}_3 , we would **only need 6 routing decision variables, which is quite a reduction with respect to the 168 arc-based routing variables!**

Underlying intuition

We could define a new set of decision variables $y_{k,p} \in \{0, 1\}$, where $y_{k,p}$ is unitary if vehicle k follows path p . With such a setting and given our definition of \mathcal{P}_1 , \mathcal{P}_2 , and \mathcal{P}_3 , we would **only need 6 routing decision variables, which is quite a reduction with respect to the 168 arc-based routing variables!**

An important consideration to make (which we will further explain in a few slides) is that each of these paths must be **feasible**, e.g., it must **start and end at the warehouse, be continuous, not exceed weight capacity, and satisfy time-windows restrictions**

Underlying intuition

We could define a new set of decision variables $y_{k,p} \in \{0, 1\}$, where $y_{k,p}$ is unitary if vehicle k follows path p . With such a setting and given our definition of \mathcal{P}_1 , \mathcal{P}_2 , and \mathcal{P}_3 , we would **only need 6 routing decision variables, which is quite a reduction with respect to the 168 arc-based routing variables!**

An important consideration to make (which we will further explain in a few slides) is that each of these paths must be **feasible**, e.g., **it must start and end at the warehouse, be continuous, not exceed weight capacity, and satisfy time-windows restrictions**

Unfortunately, because \mathcal{P}_1 does not contain path $(0, 2, 3, 4, 0)$, this variant of the model would not be able to compute the optimal solution that we previously highlighted

Underlying intuition

We could define a new set of decision variables $y_{k,p} \in \{0, 1\}$, where $y_{k,p}$ is unitary if vehicle k follows path p . With such a setting and given our definition of \mathcal{P}_1 , \mathcal{P}_2 , and \mathcal{P}_3 , we would **only need 6 routing decision variables, which is quite a reduction with respect to the 168 arc-based routing variables!**

An important consideration to make (which we will further explain in a few slides) is that each of these paths must be **feasible**, e.g., **it must start and end at the warehouse, be continuous, not exceed weight capacity, and satisfy time-windows restrictions**

Unfortunately, because \mathcal{P}_1 does not contain path $(0, 2, 3, 4, 0)$, this variant of the model would not be able to compute the optimal solution that we previously highlighted

Before diving into column generation, let us recap the arc-based formulation of the CVRP-TW and how it can be modified to accommodate the notion of paths

CVRP-TW: arc-based formulation

$$\min \quad \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k \quad (1)$$

subject to:

$$\sum_{i \in \mathcal{C}} x_{0i}^k \leq 1 \quad \forall k \in \mathcal{K} \quad (2)$$

$$\sum_{i \in \mathcal{C}} x_{0i}^k = \sum_{i \in \mathcal{C}} x_{i0}^k \quad \forall k \in \mathcal{K} \quad (3)$$

$$\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}} x_{ic}^k = 1 \quad \forall c \in \mathcal{C} \quad (4)$$

$$\sum_{j \in \mathcal{N}} x_{ji}^k = \sum_{j \in \mathcal{N}} x_{ij}^k \quad \forall i \in \mathcal{C}, k \in \mathcal{K} \quad (5)$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k \leq Q_k \quad \forall k \in \mathcal{K} \quad (6)$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k \leq R_k \quad \forall k \in \mathcal{K} \quad (7)$$

CVRP-TW: arc-based formulation

$$t_j \geq t_i + P_i + T_{ij} - (1 - \sum_{k \in \mathcal{K}} x_{ij}^k)M \quad \forall i \in \mathcal{N}, j \in \mathcal{N}, k \in \mathcal{K} \quad (8)$$

$$E_c \leq t_c \leq L_c \quad \forall c \in \mathcal{C} \quad (9)$$

$$x_{ij}^k \in \{0, 1\} \quad \forall i \in \mathcal{N}, j \in \mathcal{N}, k \in \mathcal{K} \quad (10)$$

$$t_c \geq 0 \quad \forall c \in \mathcal{C} \quad (11)$$

CVRP-TW: arc-based formulation

$$t_j \geq t_i + P_i + T_{ij} - (1 - \sum_{k \in \mathcal{K}} x_{ij}^k)M \quad \forall i \in \mathcal{N}, j \in \mathcal{N}, k \in \mathcal{K} \quad (8)$$

$$E_c \leq t_c \leq L_c \quad \forall c \in \mathcal{C} \quad (9)$$

$$x_{ij}^k \in \{0, 1\} \quad \forall i \in \mathcal{N}, j \in \mathcal{N}, k \in \mathcal{K} \quad (10)$$

$$t_c \geq 0 \quad \forall c \in \mathcal{C} \quad (11)$$

where Eq. 1 aims at minimizing the overall traveled distance, Eqs. 2-3 force a vehicle to return to the depot if used, Eq. 4 imposes that each customer must be visited, Eq. 5 ensures conservation of flow, Eqs. 6-7 enforce, respectively, maximum capacity and range restrictions, Eq. 8 imposes time precedence constraints, and Eq. 9 ensures that customers are visited within their time-window. Finally, Eqs. 10-11 define the nature of the decision variables

CVRP-TW: path-based formulation

Let us now try to write the equivalent formulation based on the notion of paths \mathcal{P}_k we previously introduced.

CVRP-TW: path-based formulation

Let us now try to write the equivalent formulation based on the notion of paths \mathcal{P}_k we previously introduced. Let us also define C_{kp} the cost of path p of vehicle k . For example, the first path for vehicle 2 is $(0, 5, 6, 7, 1, 0)$, whose cost is $D_{0,5} + D_{5,6} + D_{6,7} + D_{7,1} + D_{1,0}$.

CVRP-TW: path-based formulation

Let us now try to write the equivalent formulation based on the notion of paths \mathcal{P}_k we previously introduced. Let us also define C_{kp} the cost of path p of vehicle k . For example, the first path for vehicle 2 is $(0, 5, 6, 7, 1, 0)$, whose cost is $D_{0,5} + D_{5,6} + D_{6,7} + D_{7,1} + D_{1,0}$. Finally, let us define \mathcal{P}_{kc} **the subsets of paths of vehicle k that contain customer $c \in \mathcal{C}$.**

CVRP-TW: path-based formulation

Let us now try to write the equivalent formulation based on the notion of paths \mathcal{P}_k we previously introduced. Let us also define C_{kp} the cost of path p of vehicle k . For example, the first path for vehicle 2 is $(0, 5, 6, 7, 1, 0)$, whose cost is $D_{0,5} + D_{5,6} + D_{6,7} + D_{7,1} + D_{1,0}$. Finally, let us define \mathcal{P}_{kc} **the subsets of paths of vehicle k that contain customer $c \in \mathcal{C}$** . We can now re-define our problem as follows

CVRP-TW: path-based formulation

Let us now try to write the equivalent formulation based on the notion of paths \mathcal{P}_k we previously introduced. Let us also define C_{kp} the cost of path p of vehicle k . For example, the first path for vehicle 2 is $(0, 5, 6, 7, 1, 0)$, whose cost is $D_{0,5} + D_{5,6} + D_{6,7} + D_{7,1} + D_{1,0}$. Finally, let us define \mathcal{P}_{kc} **the subsets of paths of vehicle k that contain customer $c \in \mathcal{C}$** . We can now re-define our problem as follows

$$\min \quad \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k} C_{kp} y_{kp} \quad (12)$$

subject to:

$$\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_{kc}} y_{kp} = 1 \quad \forall c \in \mathcal{C} \quad (13)$$

$$\sum_{p \in \mathcal{P}_k} y_{kp} \leq 1 \quad \forall k \in \mathcal{K} \quad (14)$$

$$y_{kp} \in \{0, 1\} \quad \forall k \in \mathcal{K}, p \in \mathcal{P}_k \quad (15)$$

CVRP-TW: path-based formulation

Note how the formulation seems much simpler now.

CVRP-TW: path-based formulation

Note how the formulation seems much simpler now. The objective still aims at minimizing the traveled distance, while Eq. 13 is the equivalent of Eq. 4 and ensures each customer is visited exactly once.

CVRP-TW: path-based formulation

Note how the formulation seems much simpler now. The objective still aims at minimizing the traveled distance, while Eq. 13 is the equivalent of Eq. 4 and ensures each customer is visited exactly once. Among all paths that visit customer $c \in \mathcal{C}$, it forces the solution to choose exactly one.

CVRP-TW: path-based formulation

Note how the formulation seems much simpler now. The objective still aims at minimizing the traveled distance, while Eq. 13 is the equivalent of Eq. 4 and ensures each customer is visited exactly once. Among all paths that visit customer $c \in \mathcal{C}$, it forces the solution to choose exactly one. Eq. 14 ensures each vehicle is assigned at most one path, while Eq. 15 defines the nature of the decision variables

CVRP-TW: path-based formulation

Note how the formulation seems much simpler now. The objective still aims at minimizing the traveled distance, while Eq. 13 is the equivalent of Eq. 4 and ensures each customer is visited exactly once. Among all paths that visit customer $c \in \mathcal{C}$, it forces the solution to choose exactly one. Eq. 14 ensures each vehicle is assigned at most one path, while Eq. 15 defines the nature of the decision variables

Let us focus on Eq. 13 and on customer 6, in particular. Both paths of vehicle 2 visit it, together with the second path of vehicle 3. Hence we should write

CVRP-TW: path-based formulation

Note how the formulation seems much simpler now. The objective still aims at minimizing the traveled distance, while Eq. 13 is the equivalent of Eq. 4 and ensures each customer is visited exactly once. Among all paths that visit customer $c \in \mathcal{C}$, it forces the solution to choose exactly one. Eq. 14 ensures each vehicle is assigned at most one path, while Eq. 15 defines the nature of the decision variables

Let us focus on Eq. 13 and on customer 6, in particular. Both paths of vehicle 2 visit it, together with the second path of vehicle 3. Hence we should write

$$y_{2,1} + y_{2,2} + y_{3,2} = 1 \tag{16}$$

CVRP-TW: path-based formulation

Note how the formulation seems much simpler now. The objective still aims at minimizing the traveled distance, while Eq. 13 is the equivalent of Eq. 4 and ensures each customer is visited exactly once. Among all paths that visit customer $c \in \mathcal{C}$, it forces the solution to choose exactly one. Eq. 14 ensures each vehicle is assigned at most one path, while Eq. 15 defines the nature of the decision variables

Let us focus on Eq. 13 and on customer 6, in particular. Both paths of vehicle 2 visit it, together with the second path of vehicle 3. Hence we should write

$$y_{2,1} + y_{2,2} + y_{3,2} = 1 \tag{16}$$

Let us now assume we decided to add a third path for vehicle 1, namely $(0, 2, 3, 4, 0)$. What changes take place in the model?

CVRP-TW: path-based formulation

We need an **additional variable** $y_{1,3}$. In orange are highlighted all the changes to the original model

CVRP-TW: path-based formulation

We need an **additional variable** $y_{1,3}$. In orange are highlighted all the changes to the original model

$$\min \quad C_{1,1}y_{1,1} + C_{1,2}y_{1,2} + C_{2,1}y_{2,1} + C_{2,2}y_{2,2} + C_{3,1}y_{3,1} + C_{3,2}y_{3,2} + \textcolor{red}{C_{1,3}y_{1,3}} \quad (17)$$

$$y_{1,1} + y_{1,2} + \textcolor{red}{y_{1,3}} \leq 1 \quad (18)$$

$$y_{3,1} + \textcolor{red}{y_{1,3}} = 1 \quad (19)$$

$$y_{2,2} + \textcolor{red}{y_{1,3}} = 1 \quad (20)$$

$$y_{1,1} + y_{3,2} + \textcolor{red}{y_{1,3}} = 1 \quad (21)$$

CVRP-TW: path-based formulation

We need an **additional variable** $y_{1,3}$. In orange are highlighted all the changes to the original model

$$\min \quad C_{1,1}y_{1,1} + C_{1,2}y_{1,2} + C_{2,1}y_{2,1} + C_{2,2}y_{2,2} + C_{3,1}y_{3,1} + C_{3,2}y_{3,2} + \textcolor{red}{C_{1,3}y_{1,3}} \quad (17)$$

$$y_{1,1} + y_{1,2} + \textcolor{red}{y_{1,3}} \leq 1 \quad (18)$$

$$y_{3,1} + \textcolor{red}{y_{1,3}} = 1 \quad (19)$$

$$y_{2,2} + \textcolor{red}{y_{1,3}} = 1 \quad (20)$$

$$y_{1,1} + y_{3,2} + \textcolor{red}{y_{1,3}} = 1 \quad (21)$$

where in Eq. 17 there is an additional term in the objective function. Eq. 18 states that now vehicle 1 can choose from 3 options, while Eqs. 19-21 states that there are now 2,2, and 3 possible paths visiting customer 2, 3, and 4, respectively

CVRP-TW: path-based formulation

We need an **additional variable** $y_{1,3}$. In orange are highlighted all the changes to the original model

$$\min \quad C_{1,1}y_{1,1} + C_{1,2}y_{1,2} + C_{2,1}y_{2,1} + C_{2,2}y_{2,2} + C_{3,1}y_{3,1} + C_{3,2}y_{3,2} + \textcolor{red}{C_{1,3}y_{1,3}} \quad (17)$$

$$y_{1,1} + y_{1,2} + \textcolor{red}{y_{1,3}} \leq 1 \quad (18)$$

$$y_{3,1} + \textcolor{red}{y_{1,3}} = 1 \quad (19)$$

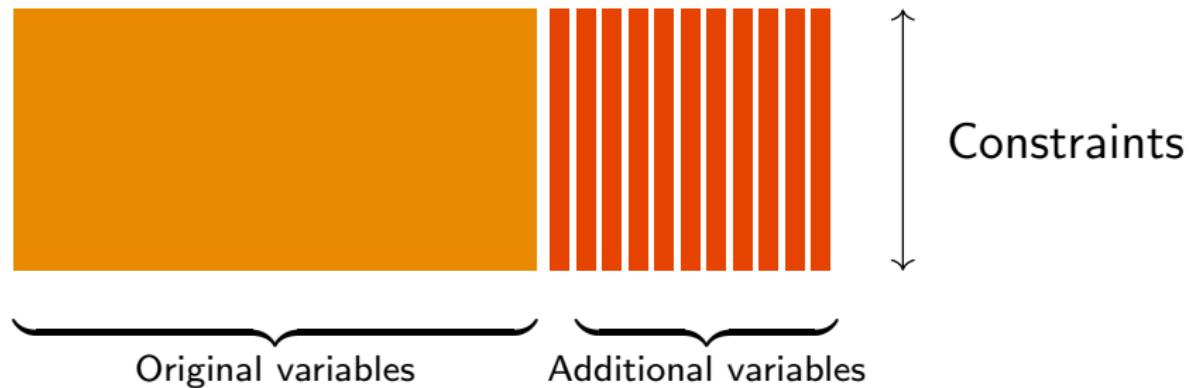
$$y_{2,2} + \textcolor{red}{y_{1,3}} = 1 \quad (20)$$

$$y_{1,1} + y_{3,2} + \textcolor{red}{y_{1,3}} = 1 \quad (21)$$

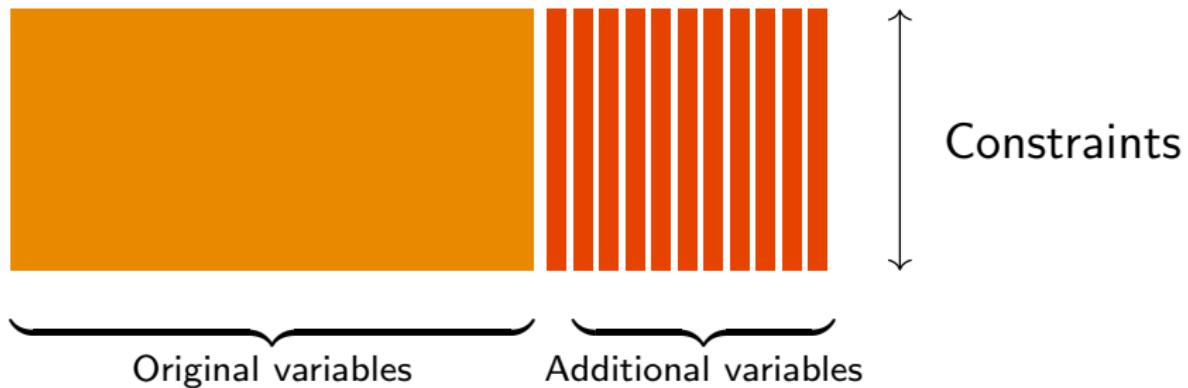
where in Eq. 17 there is an additional term in the objective function. Eq. 18 states that now vehicle 1 can choose from 3 options, while Eqs. 19-21 states that there are now 2,2, and 3 possible paths visiting customer 2, 3, and 4, respectively

We do not need to add constraints. We only need to modify some of the existing ones when adding the new variable

Why is the algorithm called Column Generation (CG)?



Why is the algorithm called Column Generation (CG)?



The number of constraints (**rows**) remains constant. Capacity and other types of constraints are already defined with our original variables. Adding more variables (**columns**) adds terms to each constraint, but does not call for the generation of new constraints.

CG algorithm in nutshell

Our overarching goal is to **translate an arc-based formulation into a path-based** and solve it by finding “good” paths

CG algorithm in nutshell

Our overarching goal is to **translate an arc-based formulation into a path-based** and solve it by finding “good” paths

Unfortunately, **full enumeration of all paths is not viable**, as it would result into a model that is computationally worse than the original arc-based one

CG algorithm in nutshell

Our overarching goal is to **translate an arc-based formulation into a path-based** and solve it by finding “good” paths

Unfortunately, **full enumeration of all paths is not viable**, as it would result into a model that is computationally worse than the original arc-based one

This is where CG comes into play. The overarching algorithm can be summarized as follows

CG algorithm in nutshell

Our overarching goal is to **translate an arc-based formulation into a path-based** and solve it by finding “good” paths

Unfortunately, **full enumeration of all paths is not viable**, as it would result into a model that is computationally worse than the original arc-based one

This is where CG comes into play. The overarching algorithm can be summarized as follows

- ① **initialize the path-based model (also called Restricted Master Problem (RMP)) with some initial paths** (decision variables) and solve it

CG algorithm in nutshell

Our overarching goal is to **translate an arc-based formulation into a path-based** and solve it by finding “good” paths

Unfortunately, **full enumeration of all paths is not viable**, as it would result into a model that is computationally worse than the original arc-based one

This is where CG comes into play. The overarching algorithm can be summarized as follows

- ① **initialize the path-based model (also called Restricted Master Problem (RMP)) with some initial paths** (decision variables) and solve it
- ② **use information from the solution to the RMP to compute “promising” paths** and add them to the RMP (also called **pricing problem**)

CG algorithm in nutshell

Our overarching goal is to **translate an arc-based formulation into a path-based** and solve it by finding “good” paths

Unfortunately, **full enumeration of all paths is not viable**, as it would result into a model that is computationally worse than the original arc-based one

This is where CG comes into play. The overarching algorithm can be summarized as follows

- ① **initialize the path-based model (also called Restricted Master Problem (RMP)) with some initial paths** (decision variables) and solve it
- ② **use information from the solution to the RMP to compute “promising” paths** and add them to the RMP (also called **pricing problem**)
- ③ keep doing so **until there is a need for it**

CG algorithm in nutshell

We already showed an example of RMP for the CVRP-TW, but did not elaborate on how to define the initial set of paths

CG algorithm in nutshell

We already showed an example of RMP for the CVRP-TW, but did not elaborate on how to define the initial set of paths

They can generally be generated **using a simple heuristic, such as a greedy heuristic**. In the CVRP-TW case, we could for example send a vehicle to the closest customer from the origin, then to the closest customer to the one just visited etc. while considering that all constraints (range, demand, time-windows) are not violated. Then, we remove the served customers and do the same with the next vehicle etc.

CG algorithm in nutshell

We already showed an example of RMP for the CVRP-TW, but did not elaborate on how to define the initial set of paths

They can generally be generated **using a simple heuristic, such as a greedy heuristic**. In the CVRP-TW case, we could for example send a vehicle to the closest customer from the origin, then to the closest customer to the one just visited etc. while considering that all constraints (range, demand, time-windows) are not violated. Then, we remove the served customers and do the same with the next vehicle etc.

In general, **the better the initial set of paths is, the faster the overall convergence should be. It is also important to ensure feasibility**. If the set of paths never passes through a customer $c \in \mathcal{C}$, then the initial RMP is infeasible as constraint **13** cannot be satisfied

Dual values

To understand how to generate new promising paths, we must first take a step back and review the concept of **dual values**

Dual values

To understand how to generate new promising paths, we must first take a step back and review the concept of **dual values**

Given a Linear Program (LP)

$$\max \quad c^T x \tag{22}$$

subject to:

$$Ax \leq b \tag{23}$$

$$x \geq 0 \tag{24}$$

Dual values

To understand how to generate new promising paths, we must first take a step back and review the concept of **dual values**

Given a Linear Program (LP)

$$\max \quad c^T x \tag{22}$$

subject to:

$$Ax \leq b \tag{23}$$

$$x \geq 0 \tag{24}$$

Every constraint in 23 is mapped by a variable (dual value or shadow price y) in the dual problem such that

Dual values

the **reduced costs** of the original decision variables of the problem (the values appearing in the objective row of the simplex tableau) are

$$c_r = c - y^T A \quad (25)$$

Dual values

the **reduced costs** of the original decision variables of the problem (the values appearing in the objective row of the simplex tableau) are

$$c_r = c - y^T A \quad (25)$$

The reduced cost of a decision variable x_j is $c_r^j = c_j - y^T A_j$ (with A_j the j-th column of the constraint matrix) and measures **how much the objective function would improve (or worsen) if the variable x_j were to enter the solution (from a non-basic state)**

Dual values

the **reduced costs** of the original decision variables of the problem (the values appearing in the objective row of the simplex tableau) are

$$c_r = c - y^T A \quad (25)$$

The reduced cost of a decision variable x_j is $c_r^j = c_j - y^T A_j$ (with A_j the j-th column of the constraint matrix) and measures **how much the objective function would improve (or worsen) if the variable x_j were to enter the solution (from a non-basic state)**

For a currently **non-basic variable x_j**

- $c_r^j > 0$ in a max problem implies improving the objective if x_j becomes basic (whereas $c_r^j < 0$ implies worsening the objective if x_j becomes basic)
- $c_r^j < 0$ in a min problem implies improving the objective if x_j becomes basic (whereas $c_r^j > 0$ implies worsening the objective if x_j becomes basic)

Dual values

This is **consistent with the simplex method algorithm** and how it decides, at every iteration, which non-basic variable should become basic. Let us consider the following example

$$\max Z = 2x_1 + 5x_2 \quad (26)$$

subject to:

$$x_1 \leq 8 \quad (27)$$

$$x_2 \leq 6 \quad (28)$$

$$3x_1 + 4x_2 \leq 36 \quad (29)$$

$$x_1, x_2 \geq 0 \quad (30)$$

which is solved in 2 iterations

Dual Values

by first making x_2 basic (most-negative coefficient)

	Z	x_1	x_2	x_3	x_4	x_5	R.H.S.
	1	-2	-5	0	0	0	0
(x_3)	0	1	0	1	0	0	8
(x_4)	0	0	1	0	1	0	6
(x_5)	0	3	4	0	0	1	36

Dual Values

and then by making x_1 basic (most-negative coefficient)

Z	x_1	x_2	x_3	x_4	x_5	R.H.S.
1	-2	0	0	5	0	30
(x_3)	0	1	0	1	0	8
(x_2)	0	0	1	0	1	6
(x_5)	0	3	0	0	-4	12

Dual Values

so that the final tableau is

Z	x_1	x_2	x_3	x_4	x_5	R.H.S.
1	0	0	0	$13/3$	$2/3$	38
(x_3)	0	0	1	$4/3$	$-1/3$	4
(x_2)	0	1	0	1	0	6
(x_1)	0	1	0	0	$-4/3$	4

Dual Values

Because all the reduced cost coefficients in the final tableau are non-negative, the current solution $(x_1, x_2) = (4, 6)$ is optimal

Dual Values

Because all the reduced cost coefficients in the final tableau are non-negative, the current solution $(x_1, x_2) = (4, 6)$ is optimal

Note that we mentioned that for a max problem we look for **non-basic decision variables with strictly positive reduced cost and, if more than one, exists, we select the one with the highest c_r^j .**

Dual Values

Because all the reduced cost coefficients in the final tableau are non-negative, the current solution $(x_1, x_2) = (4, 6)$ is optimal

Note that we mentioned that for a max problem we look for **non-basic decision variables with strictly positive reduced cost and, if more than one, exists, we select the one with the highest c_r^j** . Notwithstanding, in the previous example we selected the current non-basic variable with the most negative coefficient in the objective row!

Dual Values

Because all the reduced cost coefficients in the final tableau are non-negative, the current solution $(x_1, x_2) = (4, 6)$ is optimal

Note that we mentioned that for a max problem we look for **non-basic decision variables with strictly positive reduced cost and, if more than one, exists, we select the one with the highest c_r^j** . Notwithstanding, in the previous example we selected the current non-basic variable with the most negative coefficient in the objective row!

This is correct, as in the simplex method we re-write $Z = c^T x$ as $Z - c^T x = 0$ and hence the coefficients in the objective row are the negative of the reduced costs, i.e., $-c_r = y^T A - c$. Hence, for a max problem we look for the most negative value (as that maps the most positive reduced cost)

Pricing Problem

We can use the same intuition to define our pricing problem in the context of CG

Pricing Problem

We can use the same intuition to define our pricing problem in the context of CG

Given the current RMP, we can **solve its linear relaxation (otherwise dual values cannot be determined)** and compute the dual values for every set of constraints where path decision variables appear

Pricing Problem

We can use the same intuition to define our pricing problem in the context of CG

Given the current RMP, we can **solve its linear relaxation (otherwise dual values cannot be determined)** and compute the dual values for every set of constraints where path decision variables appear

Then, we can use this information to **assess if there are new feasible paths (decision variables) with a strictly positive reduced cost (for a max problem)** or strictly negative reduced cost (for a min problem) that should be added to the RMP

Pricing Problem

We can use the same intuition to define our pricing problem in the context of CG

Given the current RMP, we can **solve its linear relaxation (otherwise dual values cannot be determined)** and compute the dual values for every set of constraints where path decision variables appear

Then, we can use this information to **assess if there are new feasible paths (decision variables) with a strictly positive reduced cost (for a max problem)** or strictly negative reduced cost (for a min problem) that should be added to the RMP

Let us consider again our CVRP-TW problem

Pricing Problem

The “**original” cost of a path** for a given vehicle k in our CVRP-TW problem is

Pricing Problem

The “**original” cost of a path** for a given vehicle k in our CVRP-TW problem is

$$c = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k \quad (31)$$

Pricing Problem

The “**original” cost of a path** for a given vehicle k in our CVRP-TW problem is

$$c = \sum_{i \in N} \sum_{j \in N} D_{ij} x_{ij}^k \quad (31)$$

i.e., it is the summation of the distances between all nodes transversed by the vehicle (note that here **we need again arc-based decision variables to compute such a path and cost**)

Pricing Problem

The “**original” cost of a path** for a given vehicle k in our CVRP-TW problem is

$$c = \sum_{i \in N} \sum_{j \in N} D_{ij} x_{ij}^k \quad (31)$$

i.e., it is the summation of the distances between all nodes transversed by the vehicle (note that here **we need again arc-based decision variables to compute such a path and cost**)

Now, considering the path-based formulation of the RMP we defined a few slides back, we have two sets of dual

- from Eq. 13 we get π_c , **a dual value for every customer**
- from Eq. 14 we get λ_k , **a dual value for every vehicle**

Pricing Problem

This allows us to write the **reduced cost of a path for a given vehicle k** in our CVRP-TW problem as

Pricing Problem

This allows us to write the **reduced cost of a path for a given vehicle k** in our CVRP-TW problem as

$$c_r = \underbrace{\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k}_{c} - \underbrace{\sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} 1 \times \pi_c x_{ic}^k}_{A} - \underbrace{1 \times \lambda_k}_{B} \quad (32)$$

Pricing Problem

This allows us to write the **reduced cost of a path for a given vehicle k** in our CVRP-TW problem as

$$c_r = \underbrace{\sum_{i \in N} \sum_{j \in N} D_{ij} x_{ij}^k}_{c} - \underbrace{\sum_{i \in N} \sum_{c \in C} 1 \times \pi_c x_{ic}^k}_{A} - \underbrace{1 \times \lambda_k}_{B} \quad (32)$$

where we also made explicit that all duals are multiplied by the original a_{ij} coefficient of the constraint matrix (1 in this case). We have that

- A is the **contribution of visiting (or not) customers** and depends on the actual routing choices
- B is the **contribution of using vehicle k and is independent from the routing. Hence, λ_k is just a number and could be omitted from the optimization** of the pricing problem. As we are looking for a negative reduced cost (it is a min problem), we must consider anyhow the contribution of λ_k in the end

Pricing Problem

The **pricing problem** is indeed an optimization problem on its own.
Hence, our objective is to minimize the reduced cost

Pricing Problem

The **pricing problem** is indeed an optimization problem on its own.
Hence, our objective is to minimize the reduced cost

$$\min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k - \sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} \pi_c x_{ic}^k - \lambda_k \quad (33)$$

Pricing Problem

The **pricing problem is indeed an optimization problem on its own**. Hence, our objective is to minimize the reduced cost

$$\min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k - \sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} \pi_c x_{ic}^k - \lambda_k \quad (33)$$

by **properly selecting decision variables x_{ij}^k such that the resulting path is feasible**. Do you remember when we mentioned that the formulation of the RMP was very simple?

Pricing Problem

The **pricing problem is indeed an optimization problem on its own**. Hence, our objective is to minimize the reduced cost

$$\min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k - \sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} \pi_c x_{ic}^k - \lambda_k \quad (33)$$

by **properly selecting decision variables x_{ij}^k such that the resulting path is feasible**. Do you remember when we mentioned that the formulation of the RMP was very simple? This is because **most of the problem complexity is moved to the pricing problem when looking for a feasible path**.

Pricing Problem

The **pricing problem is indeed an optimization problem on its own**. Hence, our objective is to minimize the reduced cost

$$\min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k - \sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} \pi_c x_{ic}^k - \lambda_k \quad (33)$$

by **properly selecting decision variables x_{ij}^k such that the resulting path is feasible**. Do you remember when we mentioned that the formulation of the RMP was very simple? This is because **most of the problem complexity is moved to the pricing problem when looking for a feasible path**. For vehicle k , the full pricing problem is

Pricing Problem

$$\min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k - \sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} \pi_c x_{ic}^k - \lambda_k \quad (34)$$

subject to:

$$\sum_{i \in \mathcal{C}} x_{0i}^k \leq 1 \quad (35)$$

$$\sum_{i \in \mathcal{C}} x_{0i}^k = \sum_{i \in \mathcal{C}} x_{i0}^k \quad (36)$$

$$\sum_{j \in \mathcal{N}} x_{ji}^k = \sum_{j \in \mathcal{N}} x_{ij}^k \quad \forall i \in \mathcal{C} \quad (37)$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k \leq Q_k \quad (38)$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} D_{ij} x_{ij}^k \leq R_k \quad (39)$$

Pricing Problem

$$t_j \geq t_i + P_i + T_{ij} - (1 - \sum_{k \in \mathcal{K}} x_{ij}^k)M \quad \forall i \in \mathcal{N}, j \in \mathcal{N} \quad (40)$$

$$E_c \leq t_c \leq L_c \quad \forall c \in \mathcal{C} \quad (41)$$

$$x_{ij}^k \in \{0, 1\} \quad \forall i \in \mathcal{N}, j \in \mathcal{N} \quad (42)$$

$$t_c \geq 0 \quad \forall c \in \mathcal{C} \quad (43)$$

Pricing Problem

$$t_j \geq t_i + P_i + T_{ij} - (1 - \sum_{k \in \mathcal{K}} x_{ij}^k)M \quad \forall i \in \mathcal{N}, j \in \mathcal{N} \quad (40)$$

$$E_c \leq t_c \leq L_c \quad \forall c \in \mathcal{C} \quad (41)$$

$$x_{ij}^k \in \{0, 1\} \quad \forall i \in \mathcal{N}, j \in \mathcal{N} \quad (42)$$

$$t_c \geq 0 \quad \forall c \in \mathcal{C} \quad (43)$$

Note that, as we are solving for a specific vehicle k , index k in decision variables x_{ij}^k is constant and could formally be omitted

Pricing Problem

$$t_j \geq t_i + P_i + T_{ij} - (1 - \sum_{k \in \mathcal{K}} x_{ij}^k)M \quad \forall i \in \mathcal{N}, j \in \mathcal{N} \quad (40)$$

$$E_c \leq t_c \leq L_c \quad \forall c \in \mathcal{C} \quad (41)$$

$$x_{ij}^k \in \{0, 1\} \quad \forall i \in \mathcal{N}, j \in \mathcal{N} \quad (42)$$

$$t_c \geq 0 \quad \forall c \in \mathcal{C} \quad (43)$$

Note that, as we are solving for a specific vehicle k , index k in decision variables x_{ij}^k is constant and could formally be omitted

In such a pricing problem, we take care of ensuring the path is continuous, of capacity and range limitations and time-windows. **Note that we do not force every customer to be visited, as generally this is not the case for a single path (consistently, in the RMP we ensure that we combine paths in a way that all customers are served)**

Pricing Problem

Note that for a heterogeneous fleet we need to solve $|\mathcal{K}|$ pricing problems, as every vehicle might have different parameters Q_k and R_k and dual values λ_k are generally vehicle-specific

Pricing Problem

Note that for a heterogeneous fleet we need to solve $|\mathcal{K}|$ pricing problems, as every vehicle might have different parameters Q_k and R_k and dual values λ_k are generally vehicle-specific

Conversely, **for a homogeneous fleet, we can drop the index k and solve the pricing problem once and add the resulting computed path to every path set \mathcal{P}_k , as such a path is feasible for every vehicle in the fleet (being all vehicles equal)**

The full CG algorithm

In essence, we should start with an **initial RMP, solve it and compute the duals**. Use them to solve the pricing problem, find suitable paths to be added to the RMP, solve the RMP again and **recompute the duals** until the pricing problem yields paths with a positive reduced cost

The full CG algorithm

In essence, we should start with an **initial RMP, solve it and compute the duals**. Use them to solve the pricing problem, find suitable paths to be added to the RMP, solve the RMP again and **recompute the duals** until the pricing problem yields paths with a positive reduced cost

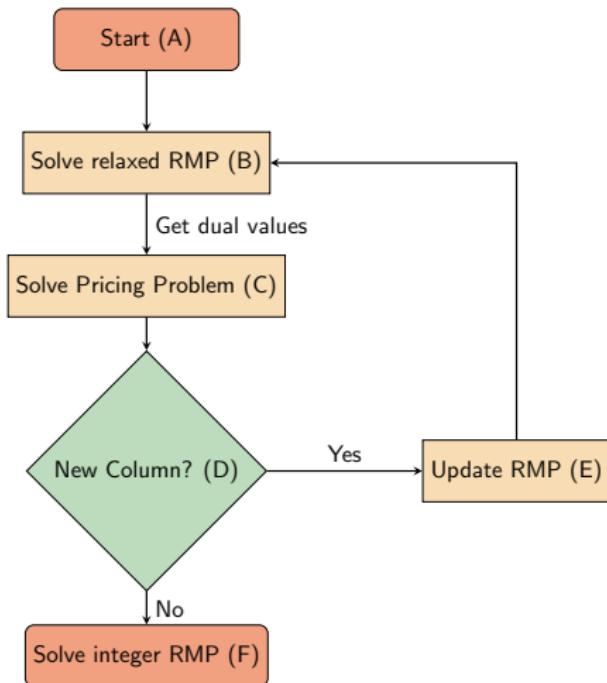
This final step ensures that the **linear relaxation of the final RMP has been solved to optimality** (akin to the simplex method). As either we choose or not a path in the path-based formulation (whereas a linear relaxation might allow half a path to be chosen), we need a final step.

The full CG algorithm

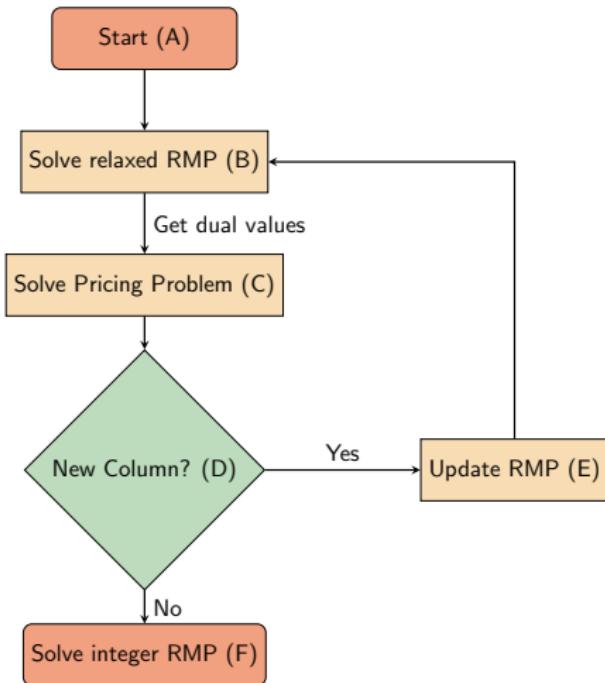
In essence, we should start with an **initial RMP, solve it and compute the duals**. Use them to solve the pricing problem, find suitable paths to be added to the RMP, solve the RMP again and **recompute the duals** until the pricing problem yields paths with a positive reduced cost

This final step ensures that the **linear relaxation of the final RMP has been solved to optimality** (akin to the simplex method). As either we choose or not a path in the path-based formulation (whereas a linear relaxation might allow half a path to be chosen), we need a final step. Once no more paths can be added, we should solve again the final RMP by reinstating the integral (binary) nature of the decision variables. **This might (and generally will) result in a degradation of the objective value**

The full CG algorithm

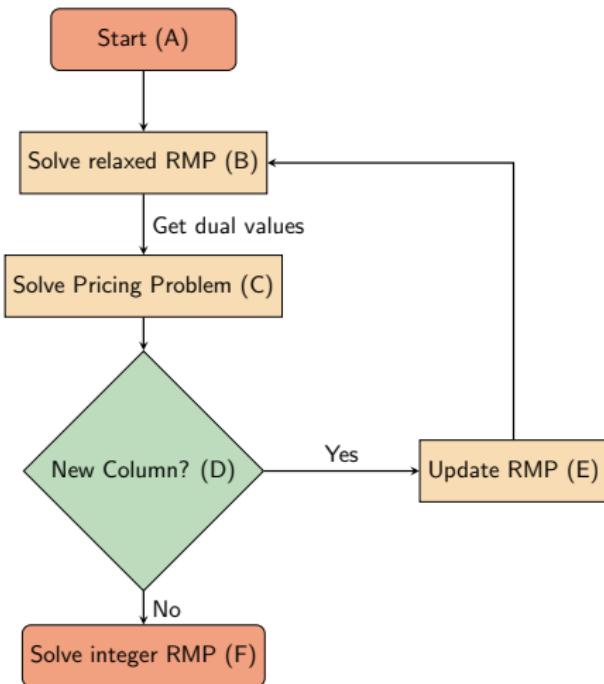


The full CG algorithm



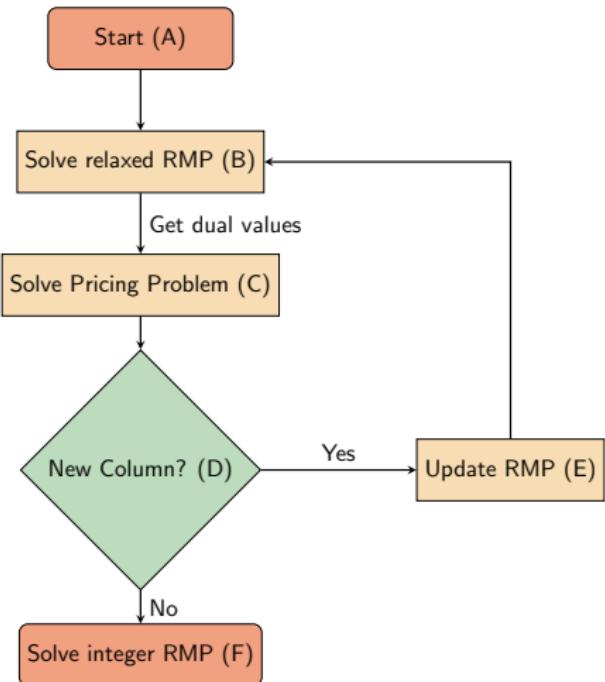
(A) is a pre-processing step where an **initial set of paths \mathcal{P} must be computed**.

The full CG algorithm



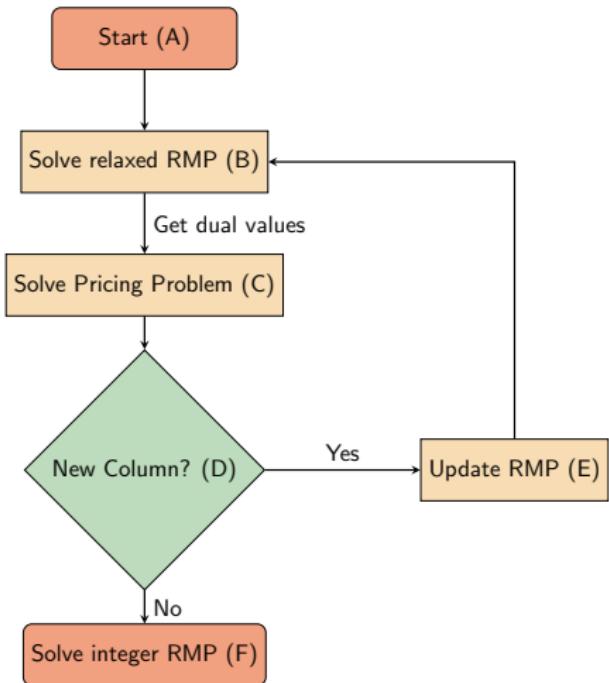
(A) is a pre-processing step where an **initial set of paths \mathcal{P} must be computed**. Then, (B), (C), (D), and (E) define altogether the **iterative process of the CG algorithm**, where as many columns as needed are added.

The full CG algorithm



(A) is a pre-processing step where an **initial set of paths \mathcal{P} must be computed**. Then, (B), (C), (D), and (E) define altogether the **iterative process of the CG algorithm**, where as many columns as needed are added. Finally, (F) solves the final RMP where **decision variables are forced to be integer to get an implementable solution**

The full CG algorithm



(A) is a pre-processing step where an **initial set of paths \mathcal{P} must be computed**. Then, (B), (C), (D), and (E) define altogether the **iterative process of the CG algorithm**, where as many columns as needed are added. Finally, (F) solves the final RMP where **decision variables are forced to be integer to get an implementable solution**

Because of (F), CG is a heuristic and is not guaranteed to provide a global optimum

A worked-out example: the 1D cutting stock problem

The **1D Cutting Stock Problem (CSP)** is a keystone optimization problem commonly encountered in manufacturing industries such as paper, metal, wood, and textiles.

A worked-out example: the 1D cutting stock problem

The **1D Cutting Stock Problem (CSP)** is a keystone optimization problem commonly encountered in **manufacturing industries such as paper, metal, wood, and textiles**. It involves determining the **most efficient way to cut large stock materials (e.g., rolls of paper, sheets of metal, or wooden planks) into smaller pieces** to meet specific customer demands while minimizing waste

A worked-out example: the 1D cutting stock problem

The **1D Cutting Stock Problem (CSP)** is a keystone optimization problem commonly encountered in **manufacturing industries such as paper, metal, wood, and textiles**. It involves determining the **most efficient way to cut large stock materials (e.g., rolls of paper, sheets of metal, or wooden planks) into smaller pieces** to meet specific customer demands while minimizing waste

In the 1D variant, we can consider **“long” 2D rolls of material where customer orders are about specific lengths of the rolls** (with the width being just a constant).

A worked-out example: the 1D cutting stock problem

The **1D Cutting Stock Problem (CSP)** is a keystone optimization problem commonly encountered in **manufacturing industries such as paper, metal, wood, and textiles**. It involves determining the **most efficient way to cut large stock materials (e.g., rolls of paper, sheets of metal, or wooden planks) into smaller pieces** to meet specific customer demands while minimizing waste

In the 1D variant, we can consider **“long” 2D rolls of material where customer orders are about specific lengths of the rolls** (with the width being just a constant). There exist more complicated 2D and even 3D variants

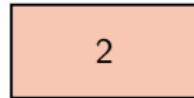
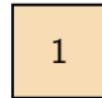
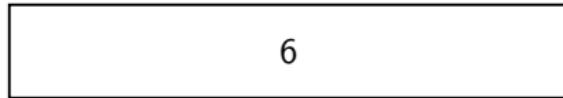
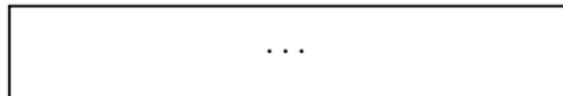
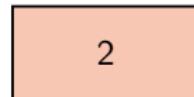
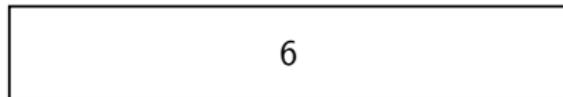
A worked-out example: the 1D cutting stock problem

The **1D Cutting Stock Problem (CSP)** is a keystone optimization problem commonly encountered in **manufacturing industries such as paper, metal, wood, and textiles**. It involves determining the **most efficient way to cut large stock materials (e.g., rolls of paper, sheets of metal, or wooden planks) into smaller pieces** to meet specific customer demands while minimizing waste

In the 1D variant, we can consider **“long” 2D rolls of material where customer orders are about specific lengths of the rolls** (with the width being just a constant). There exist more complicated 2D and even 3D variants

The CSP is often solved using CG, where paths are specific cutting patterns of a roll

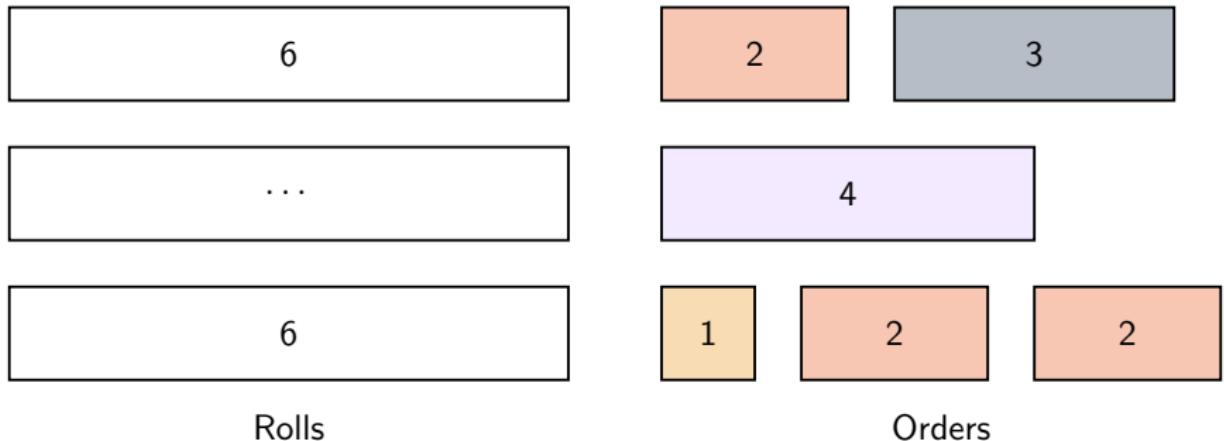
A worked-out example: the 1D cutting stock problem



Rolls

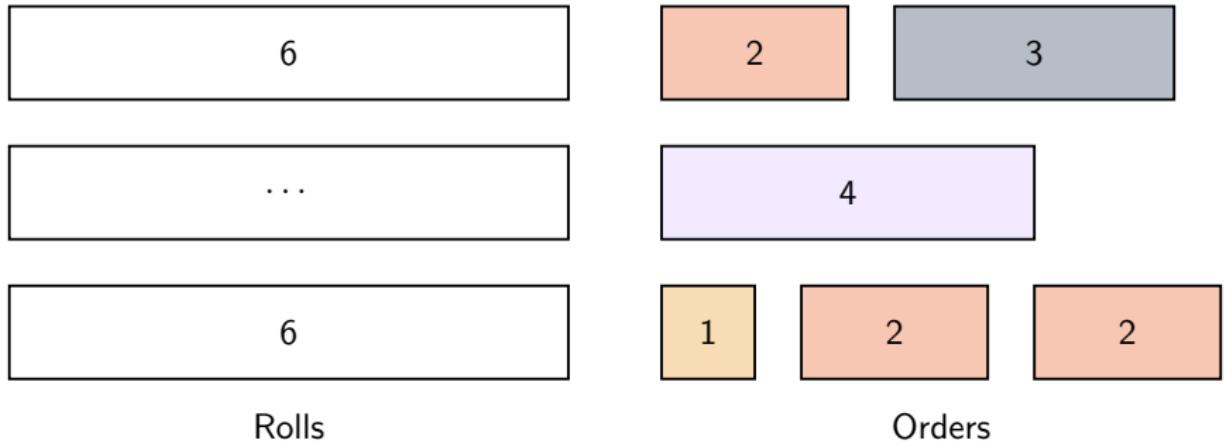
Orders

A worked-out example: the 1D cutting stock problem



How many rolls are needed to cut all the orders while minimizing waste?

A worked-out example: the 1D cutting stock problem

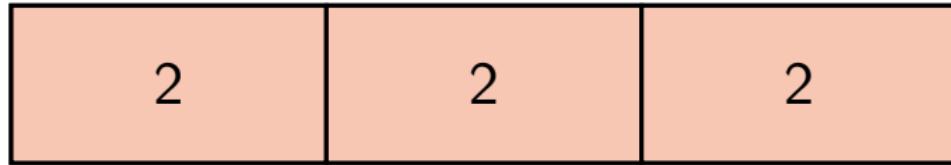
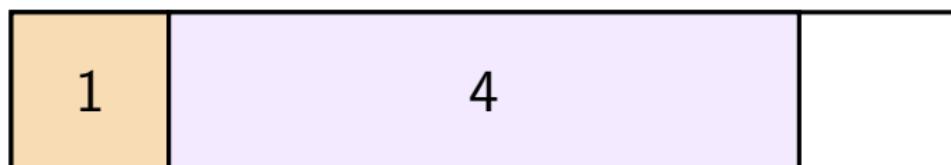


How many rolls are needed to cut all the orders while minimizing waste?

Because the overall length of all orders is $1 + 3 \times 2 + 3 + 4 = 14$ and each roll has a length of 6, we need at least $\lceil \frac{14}{6} \rceil = 3$ rolls

A worked-out example: the 1D cutting stock problem

A feasible (and optimal, albeit not unique) solution is



A worked-out example: the 1D cutting stock problem

We can model the CSP as a 1D Bin Packing Problem (1D-BPP) akin to the arc-based formulation of the CVRP-TW, where we have

A worked-out example: the 1D cutting stock problem

We can model the CSP as a 1D Bin Packing Problem (1D-BPP) akin to the arc-based formulation of the CVRP-TW, where we have

- a **set of unique orders \mathcal{O}** , each with a demanded quantity Q_o and a length L_o . In our example, $\mathcal{O} = \{1, 2, 3, 4\}$ with $Q_1 = 1$, $Q_2 = 3$, $Q_3 = 1$, and $Q_4 = 1$.

A worked-out example: the 1D cutting stock problem

We can model the CSP as a 1D Bin Packing Problem (1D-BPP) akin to the arc-based formulation of the CVRP-TW, where we have

- a **set of unique orders \mathcal{O}** , each with a demanded quantity Q_o and a length L_o . In our example, $\mathcal{O} = \{1, 2, 3, 4\}$ with $Q_1 = 1$, $Q_2 = 3$, $Q_3 = 1$, and $Q_4 = 1$. To better comply with 1D-BPP modeling, we define the set of items \mathcal{I} that contains all single orders, even if some orders are formally identical. Hence for us $|\mathcal{I}| = 6$ and L_i is the length of item i

A worked-out example: the 1D cutting stock problem

We can model the CSP as a 1D Bin Packing Problem (1D-BPP) akin to the arc-based formulation of the CVRP-TW, where we have

- a **set of unique orders \mathcal{O}** , each with a demanded quantity Q_o and a length L_o . In our example, $\mathcal{O} = \{1, 2, 3, 4\}$ with $Q_1 = 1$, $Q_2 = 3$, $Q_3 = 1$, and $Q_4 = 1$. To better comply with 1D-BPP modeling, we define the set of items \mathcal{I} that contains all single orders, even if some orders are formally identical. Hence for us $|\mathcal{I}| = 6$ and L_i is the length of item i
- a **set of rolls \mathcal{B} of length L** that act as the bins in the 1D-BPP analogy

A worked-out example: the 1D cutting stock problem

We can model the CSP as a 1D Bin Packing Problem (1D-BPP) akin to the arc-based formulation of the CVRP-TW, where we have

- a **set of unique orders \mathcal{O}** , each with a demanded quantity Q_o and a length L_o . In our example, $\mathcal{O} = \{1, 2, 3, 4\}$ with $Q_1 = 1$, $Q_2 = 3$, $Q_3 = 1$, and $Q_4 = 1$. To better comply with 1D-BPP modeling, we define the set of items \mathcal{I} that contains all single orders, even if some orders are formally identical. Hence for us $|\mathcal{I}| = 6$ and L_i is the length of item i
- a **set of rolls \mathcal{B} of length L** that act as the bins in the 1D-BPP analogy

In such a setting, we need variables $x_{i,b} \in \{0, 1\}$ (unitary if item i is assigned to bin b) and $y_b \in \{0, 1\}$ (unitary if bin b is used) and can formalize our problem as follows

A worked-out example: the 1D cutting stock problem

$$\min \sum_{b \in \mathcal{B}} y_b \quad (44)$$

subject to:

$$\sum_{b \in \mathcal{B}} x_{i,b} = 1 \quad \forall i \in \mathcal{I} \quad (45)$$

$$\sum_{i \in \mathcal{I}} L_i x_{i,b} \leq L \quad \forall b \in \mathcal{B} \quad (46)$$

$$x_{i,b} \leq y_b \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (47)$$

$$x_{i,b} \in \{0, 1\} \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (48)$$

$$y_b \in \{0, 1\} \quad \forall b \in \mathcal{B} \quad (49)$$

A worked-out example: the 1D cutting stock problem

$$\min \sum_{b \in \mathcal{B}} y_b \quad (44)$$

subject to:

$$\sum_{b \in \mathcal{B}} x_{i,b} = 1 \quad \forall i \in \mathcal{I} \quad (45)$$

$$\sum_{i \in \mathcal{I}} L_i x_{i,b} \leq L \quad \forall b \in \mathcal{B} \quad (46)$$

$$x_{i,b} \leq y_b \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (47)$$

$$x_{i,b} \in \{0, 1\} \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (48)$$

$$y_b \in \{0, 1\} \quad \forall b \in \mathcal{B} \quad (49)$$

where Eq. 44 minimizes the number of used rolls,

A worked-out example: the 1D cutting stock problem

$$\min \sum_{b \in \mathcal{B}} y_b \quad (44)$$

subject to:

$$\sum_{b \in \mathcal{B}} x_{i,b} = 1 \quad \forall i \in \mathcal{I} \quad (45)$$

$$\sum_{i \in \mathcal{I}} L_i x_{i,b} \leq L \quad \forall b \in \mathcal{B} \quad (46)$$

$$x_{i,b} \leq y_b \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (47)$$

$$x_{i,b} \in \{0, 1\} \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (48)$$

$$y_b \in \{0, 1\} \quad \forall b \in \mathcal{B} \quad (49)$$

where Eq. 44 minimizes the number of used rolls, Eq. 45 ensures that each item is assigned,

A worked-out example: the 1D cutting stock problem

$$\min \sum_{b \in \mathcal{B}} y_b \quad (44)$$

subject to:

$$\sum_{b \in \mathcal{B}} x_{i,b} = 1 \quad \forall i \in \mathcal{I} \quad (45)$$

$$\sum_{i \in \mathcal{I}} L_i x_{i,b} \leq L \quad \forall b \in \mathcal{B} \quad (46)$$

$$x_{i,b} \leq y_b \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (47)$$

$$x_{i,b} \in \{0, 1\} \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (48)$$

$$y_b \in \{0, 1\} \quad \forall b \in \mathcal{B} \quad (49)$$

where Eq. 44 minimizes the number of used rolls, Eq. 45 ensures that each item is assigned, Eq. 46 that each bin is used within capacity,

A worked-out example: the 1D cutting stock problem

$$\min \sum_{b \in \mathcal{B}} y_b \quad (44)$$

subject to:

$$\sum_{b \in \mathcal{B}} x_{i,b} = 1 \quad \forall i \in \mathcal{I} \quad (45)$$

$$\sum_{i \in \mathcal{I}} L_i x_{i,b} \leq L \quad \forall b \in \mathcal{B} \quad (46)$$

$$x_{i,b} \leq y_b \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (47)$$

$$x_{i,b} \in \{0, 1\} \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (48)$$

$$y_b \in \{0, 1\} \quad \forall b \in \mathcal{B} \quad (49)$$

where Eq. 44 minimizes the number of used rolls, Eq. 45 ensures that each item is assigned, Eq. 46 that each bin is used within capacity, and Eq. 47 “activates” a bin if an item is assigned to it.

A worked-out example: the 1D cutting stock problem

$$\min \sum_{b \in \mathcal{B}} y_b \quad (44)$$

subject to:

$$\sum_{b \in \mathcal{B}} x_{i,b} = 1 \quad \forall i \in \mathcal{I} \quad (45)$$

$$\sum_{i \in \mathcal{I}} L_i x_{i,b} \leq L \quad \forall b \in \mathcal{B} \quad (46)$$

$$x_{i,b} \leq y_b \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (47)$$

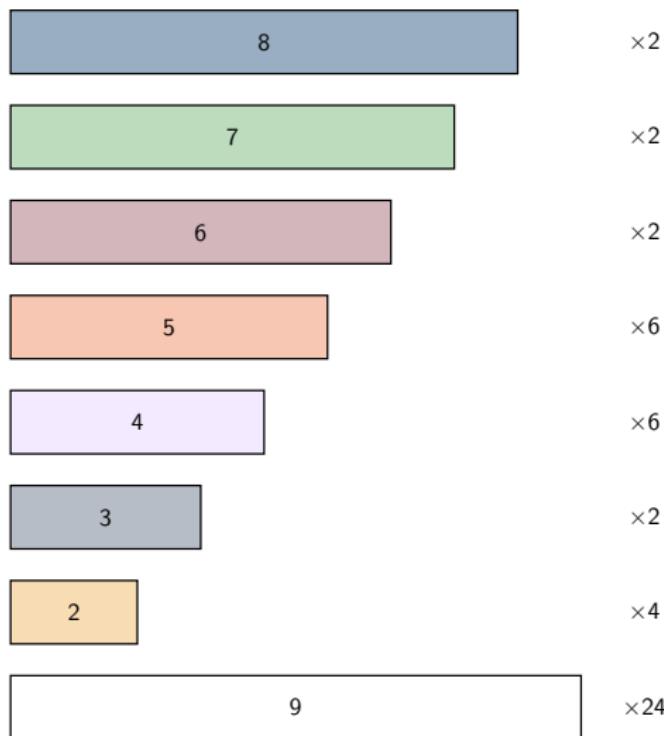
$$x_{i,b} \in \{0, 1\} \quad \forall i \in \mathcal{I}, b \in \mathcal{B} \quad (48)$$

$$y_b \in \{0, 1\} \quad \forall b \in \mathcal{B} \quad (49)$$

where Eq. 44 minimizes the number of used rolls, Eq. 45 ensures that each item is assigned, Eq. 46 that each bin is used within capacity, and Eq. 47 “activates” a bin if an item is assigned to it. Eqs. 48-49 define the nature of the decision variables

A worked-out example: the 1D cutting stock problem

Let us consider a slightly more complicated setting than before, where we have the following set of orders and bins

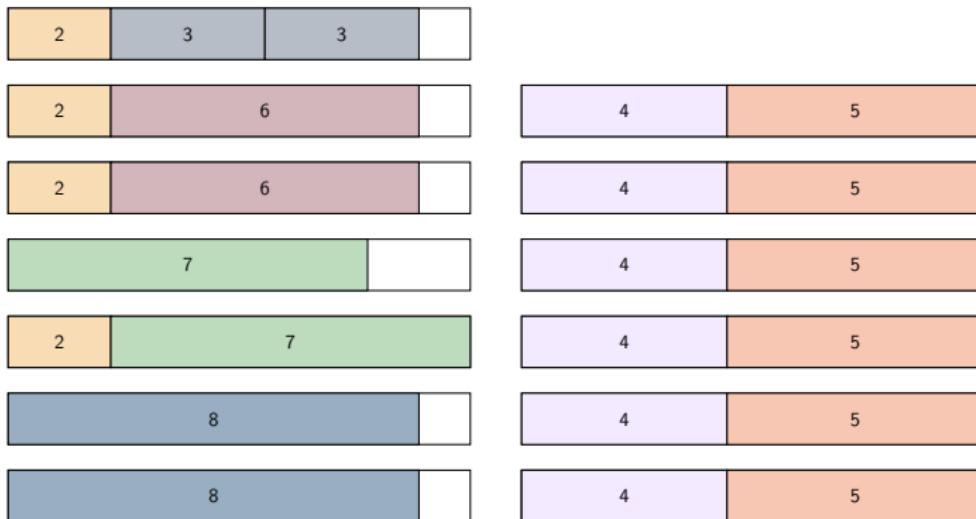


A worked-out example: the 1D cutting stock problem

Solving it with the 1D-BPP presented before, yields an optimal solution where 13 bins (rolls) are needed

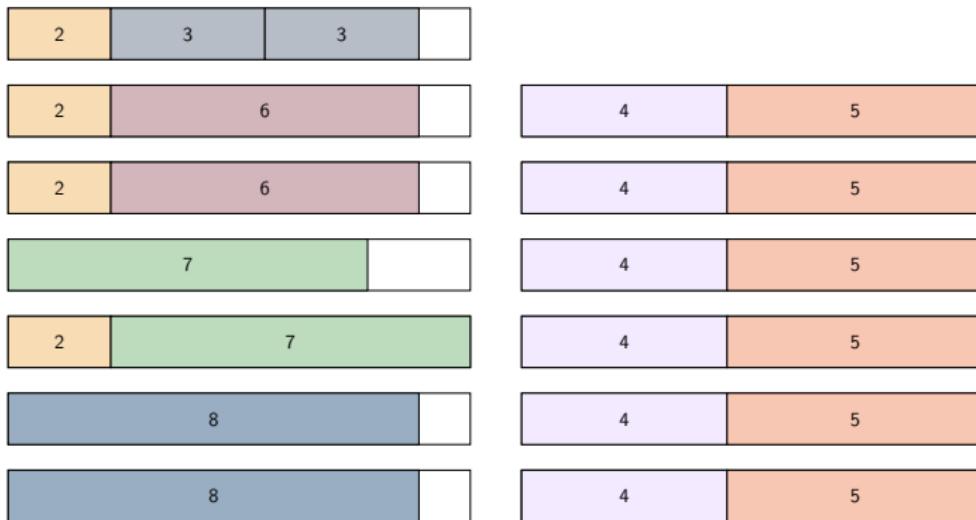
A worked-out example: the 1D cutting stock problem

Solving it with the 1D-BPP presented before, yields an optimal solution where 13 bins (rolls) are needed



A worked-out example: the 1D cutting stock problem

Solving it with the 1D-BPP presented before, yields an optimal solution where 13 bins (rolls) are needed



This was also our **lower bound** without even starting the optimization as $\lceil \frac{4 \times 2 + 2 \times 3 + 6 \times 4 + 6 \times 5 + 2 \times 6 + 2 \times 7 + 2 \times 8}{9} \rceil = 13$

A worked-out example: the 1D cutting stock problem

We now try to achieve the same value using a CG approach. To do so, we use the notation of **unique orders** \mathcal{O} we previously introduced and rely on the concept of pattern (equivalent to a path in the CVRP-TW example)

A worked-out example: the 1D cutting stock problem

We now try to achieve the same value using a CG approach. To do so, we use the notation of **unique orders** \mathcal{O} we previously introduced and rely on the concept of pattern (equivalent to a path in the CVRP-TW example)

A pattern is a combination of unique orders $o \in \mathcal{O}$ (potentially repeated) in which a roll can be cut.

A worked-out example: the 1D cutting stock problem

We now try to achieve the same value using a CG approach. To do so, we use the notation of **unique orders** \mathcal{O} we previously introduced and rely on the concept of pattern (equivalent to a path in the CVRP-TW example)

A pattern is a **combination of unique orders** $o \in \mathcal{O}$ (potentially repeated) in which a roll can be cut. For example, the following roll

A worked-out example: the 1D cutting stock problem

We now try to achieve the same value using a CG approach. To do so, we use the notation of **unique orders** \mathcal{O} we previously introduced and rely on the concept of pattern (equivalent to a path in the CVRP-TW example)

A pattern is a combination of unique orders $o \in \mathcal{O}$ (potentially repeated) in which a roll can be cut. For example, the following roll



A worked-out example: the 1D cutting stock problem

We now try to achieve the same value using a CG approach. To do so, we use the notation of **unique orders \mathcal{O}** we previously introduced and **rely on the concept of pattern (equivalent to a path in the CVRP-TW example)**

A pattern is a **combination of unique orders $o \in \mathcal{O}$ (potentially repeated) in which a roll can be cut**. For example, the following roll



defines a pattern where order 2 with length $L_2 = 2$ appears once and order 3 with length $L_3 = 3$ appears twice.

A worked-out example: the 1D cutting stock problem

We now try to achieve the same value using a CG approach. To do so, we use the notation of **unique orders** \mathcal{O} we previously introduced and rely on the concept of pattern (equivalent to a path in the CVRP-TW example)

A pattern is a combination of unique orders $o \in \mathcal{O}$ (potentially repeated) in which a roll can be cut. For example, the following roll



defines a pattern where order 2 with length $L_2 = 2$ appears once and order 3 with length $L_3 = 3$ appears twice. We will store all the patterns we have available in set \mathcal{P} and define \mathcal{P}_o the subset of patterns containing order o and N_o^p the number of times order o appears in pattern p .

A worked-out example: the 1D cutting stock problem

We now try to achieve the same value using a CG approach. To do so, we use the notation of **unique orders** \mathcal{O} we previously introduced and rely on the concept of pattern (equivalent to a path in the CVRP-TW example)

A pattern is a combination of unique orders $o \in \mathcal{O}$ (potentially repeated) in which a roll can be cut. For example, the following roll



defines a pattern where order 2 with length $L_2 = 2$ appears once and order 3 with length $L_3 = 3$ appears twice. We will store all the patterns we have available in set \mathcal{P} and define \mathcal{P}_o the subset of patterns containing order o and N_o^p the number of times order o appears in pattern p . Finally, we define integer variable x_p the number of times pattern $p \in \mathcal{P}$ is chosen in the final solution

A worked-out example: the 1D cutting stock problem

As mentioned in the general CG algorithm description, we need an initial set of patterns to start.

A worked-out example: the 1D cutting stock problem

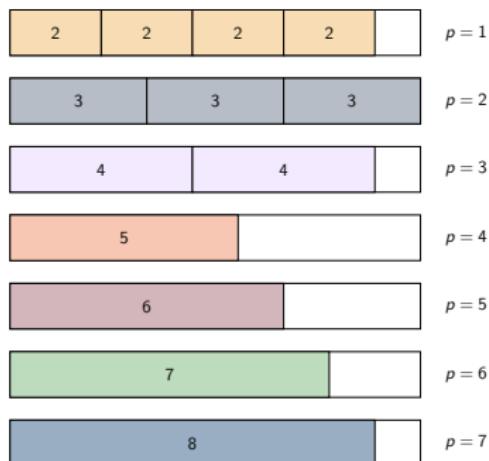
As mentioned in the general CG algorithm description, we need an initial set of patterns to start. In this case, a **viable option is to consider each order $o \in \mathcal{O}$ individually and add a copy of it to a pattern as long as there is space.**

A worked-out example: the 1D cutting stock problem

As mentioned in the general CG algorithm description, we need an initial set of patterns to start. In this case, a **viable option is to consider each order $o \in \mathcal{O}$ individually and add a copy of it to a pattern as long as there is space**. Hence, we could initialize our RMP with 7 patterns as follows

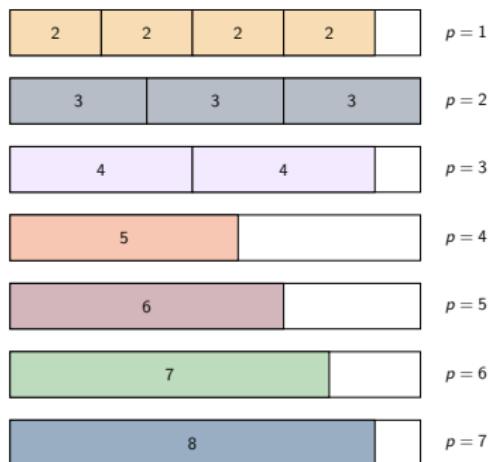
A worked-out example: the 1D cutting stock problem

As mentioned in the general CG algorithm description, we need an initial set of patterns to start. In this case, a **viable option is to consider each order $o \in \mathcal{O}$ individually and add a copy of it to a pattern as long as there is space**. Hence, we could initialize our RMP with 7 patterns as follows



A worked-out example: the 1D cutting stock problem

As mentioned in the general CG algorithm description, we need an initial set of patterns to start. In this case, a **viable option is to consider each order $o \in \mathcal{O}$ individually and add a copy of it to a pattern as long as there is space**. Hence, we could initialize our RMP with 7 patterns as follows



A couple of problems with such an initial choice

- some patterns feature a lot of **wasted space**
- related to the above, not being able to mix different orders **does not allow for flexible configurations and might lead to overproducing some orders**

A worked-out example: the 1D cutting stock problem

Let us now introduce the RMP formulation

A worked-out example: the 1D cutting stock problem

Let us now introduce the RMP formulation

$$\min \sum_{p \in \mathcal{P}} x_p \quad (50)$$

subject to:

$$\sum_{p \in \mathcal{P}_o} N_o^p x_p \geq Q_o \quad \forall o \in \mathcal{O} \quad (51)$$

$$x_p \in \mathbb{N} \quad \forall p \in \mathcal{P} \quad (52)$$

A worked-out example: the 1D cutting stock problem

Let us now introduce the RMP formulation

$$\min \sum_{p \in \mathcal{P}} x_p \quad (50)$$

subject to:

$$\sum_{p \in \mathcal{P}_o} N_o^p x_p \geq Q_o \quad \forall o \in \mathcal{O} \quad (51)$$

$$x_p \in \mathbb{N} \quad \forall p \in \mathcal{P} \quad (52)$$

Eq. 50 minimizes the number of patterns (rolls) used. Eq. 51 ensures that at least as many orders of each type are produced as requested, while Eq. 52 states the integer nature of decision variable x_p (as discussed, this restriction will be relaxed during the whole CG iterative process)

A worked-out example: the 1D cutting stock problem

Solving the relaxed RMP with the 7 patterns shown before yields an objective value of $16\frac{2}{3}$ (clearly non integer) with the following solution

A worked-out example: the 1D cutting stock problem

Solving the relaxed RMP with the 7 patterns shown before yields an objective value of $16\frac{2}{3}$ (clearly non integer) with the following solution

$$(x_1, \dots, x_7) = (1, \frac{2}{3}, 3, 6, 2, 2, 2)$$

A worked-out example: the 1D cutting stock problem

Solving the relaxed RMP with the 7 patterns shown before yields an objective value of $16\frac{2}{3}$ (clearly non integer) with the following solution

$$(x_1, \dots, x_7) = (1, \frac{2}{3}, 3, 6, 2, 2, 2)$$

We now introduce the pricing problem. As we are looking for new “promising” patterns, a reasonable decision variable to define is y_o , an integer variable that defines how many times order $o \in \mathcal{O}$ appears in the new pattern. In addition, from Eq. 51 we have $|\mathcal{O}|$ dual values λ_o , one per order

A worked-out example: the 1D cutting stock problem

Let us start with the objective. We want to minimize the reduced cost and, hopefully, find a new pattern p with a negative reduced cost.

A worked-out example: the 1D cutting stock problem

Let us start with the objective. We want to minimize the reduced cost and, hopefully, find a new pattern p with a negative reduced cost. Let us recall that the reduced cost takes the form

$$c_r = c - y^T A \quad (53)$$

A worked-out example: the 1D cutting stock problem

Let us start with the objective. We want to minimize the reduced cost and, hopefully, find a new pattern p with a negative reduced cost. Let us recall that the reduced cost takes the form

$$c_r = c - y^T A \quad (53)$$

which in our case we can write as

$$c_r = 1 - \sum_{o \in \mathcal{O}} \lambda_o N_o^p \quad (54)$$

A worked-out example: the 1D cutting stock problem

Let us start with the objective. We want to minimize the reduced cost and, hopefully, find a new pattern p with a negative reduced cost. Let us recall that the reduced cost takes the form

$$c_r = c - y^T A \quad (53)$$

which in our case we can write as

$$c_r = 1 - \sum_{o \in \mathcal{O}} \lambda_o N_o^p \quad (54)$$

We are now facing a problem though.

A worked-out example: the 1D cutting stock problem

Let us start with the objective. We want to minimize the reduced cost and, hopefully, find a new pattern p with a negative reduced cost. Let us recall that the reduced cost takes the form

$$c_r = c - y^T A \quad (53)$$

which in our case we can write as

$$c_r = 1 - \sum_{o \in \mathcal{O}} \lambda_o N_o^p \quad (54)$$

We are now facing a problem though. N_o^p is the number of times order o appears in the new pattern we are trying to compute, so it is neither known in advance nor a constant we can just plug in.

A worked-out example: the 1D cutting stock problem

Let us start with the objective. We want to minimize the reduced cost and, hopefully, find a new pattern p with a negative reduced cost. Let us recall that the reduced cost takes the form

$$c_r = c - y^T A \quad (53)$$

which in our case we can write as

$$c_r = 1 - \sum_{o \in \mathcal{O}} \lambda_o N_o^p \quad (54)$$

We are now facing a problem though. N_o^p is the number of times order o appears in the new pattern we are trying to compute, so it is neither known in advance nor a constant we can just plug in. Luckily for us, the decision variable y_o is exactly the number of times order o will appear in pattern p , and hence we can replace N_o^p with y_p . The pricing problem looks as follows

A worked-out example: the 1D cutting stock problem

$$\min \quad 1 - \sum_{o \in \mathcal{O}} \lambda_o y_o \quad (55)$$

subject to:

$$\sum_{o \in \mathcal{P}_o} L_o y_o \leq L \quad (56)$$

$$y_o \in \mathbb{N} \quad \forall o \in \mathcal{O} \quad (57)$$

A worked-out example: the 1D cutting stock problem

$$\min \quad 1 - \sum_{o \in \mathcal{O}} \lambda_o y_o \quad (55)$$

subject to:

$$\sum_{o \in \mathcal{P}_o} L_o y_o \leq L \quad (56)$$

$$y_o \in \mathbb{N} \quad \forall o \in \mathcal{O} \quad (57)$$

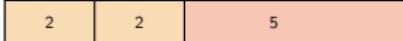
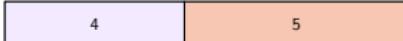
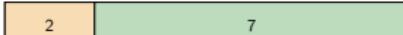
Eq. 55 minimizes the reduced cost of a pattern. Eq. 56 ensures that a roll is used within capacity, while Eq. 52 states the integer nature of decision variable y_o (differently from the RMP, here this constraint must be enforced all the time)

A worked-out example: the 1D cutting stock problem

Having now defined both the RMP and the pricing problem, and given our initial set of patterns \mathcal{P} , the CG algorithm converges in 5 iterations where the following 5 additional patterns have been identified and added

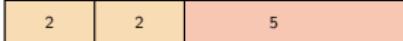
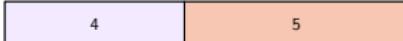
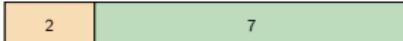
A worked-out example: the 1D cutting stock problem

Having now defined both the RMP and the pricing problem, and given our initial set of patterns \mathcal{P} , the CG algorithm converges in 5 iterations where the following 5 additional patterns have been identified and added

	$p = 8, r_c = -\frac{1}{2}$
	$p = 9, r_c = -\frac{1}{2}$
	$p = 10, r_c = -\frac{1}{3}$
	$p = 11, r_c = -\frac{1}{4}$
	$p = 12, r_c = -\frac{1}{12}$

A worked-out example: the 1D cutting stock problem

Having now defined both the RMP and the pricing problem, and given our initial set of patterns \mathcal{P} , the CG algorithm converges in 5 iterations where the following 5 additional patterns have been identified and added

	$p = 8, r_c = -\frac{1}{2}$
	$p = 9, r_c = -\frac{1}{2}$
	$p = 10, r_c = -\frac{1}{3}$
	$p = 11, r_c = -\frac{1}{4}$
	$p = 12, r_c = -\frac{1}{12}$

Solving the final RMP with the re-instated integrality restrictions on the x_p decision variables yields an objective equal to 13 and the same final solution as the one obtained with the 1D-BPP approach. More specifically we have $x_7 = 2$, $x_8 = 1$, $x_9 = 6$, $x_{10} = 2$, and $x_{11} = 2$. **While we still use 13 rolls, the actual final patterns are different from the solution obtained via the 1D-BPP (symmetrical solutions). Check!**

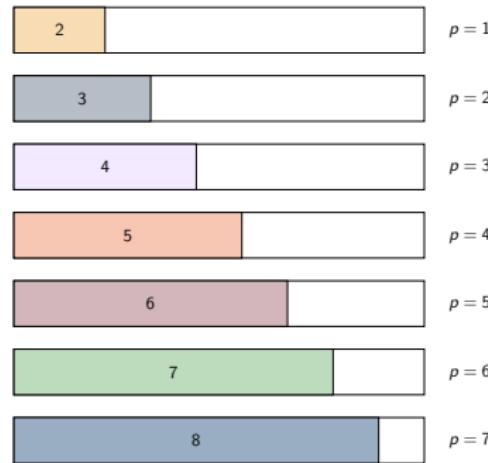
A worked-out example: the 1D cutting stock problem

You can find the implementation of this CSP here: [Github](#)

A worked-out example: the 1D cutting stock problem

You can find the implementation of this CSP here: [Github](#)

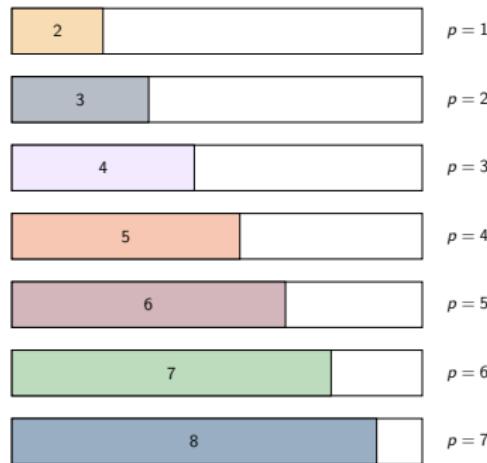
To test the effect of a different choice of initial paths, try to use this initial set



A worked-out example: the 1D cutting stock problem

You can find the implementation of this CSP here: [Github](#)

To test the effect of a different choice of initial paths, try to use this initial set



As the **initial choice** is “less efficient”, it will take more iterations (and some identified patterns are some of the better initial patterns of the previous choice)

Some final words

- the CG algorithm is useful for those problems with many combinations between decision variables (e.g., routing and packing problems), as **it allows to reduce the size of the problem**

Some final words

- the CG algorithm is useful for those problems with many combinations between decision variables (e.g., routing and packing problems), as **it allows to reduce the size of the problem**
- this entails a **model reformulation with decision variables in the form of paths (columns)**. Defining the RMP is generally straightforward

Some final words

- the CG algorithm is useful for those problems with many combinations between decision variables (e.g., routing and packing problems), as **it allows to reduce the size of the problem**
- this entails a **model reformulation with decision variables in the form of paths (columns)**. Defining the RMP is generally straight-forward
- computing new promising feasible paths is done via the pricing problem. Defining its is also straight-forward, as they are inherited from the original “arc-based” formulation. **Defining the objective value and the proper decision variables is a bit harder and requires practice (and understanding of the dual values!)**

Some final words

- the CG algorithm is useful for those problems with many combinations between decision variables (e.g., routing and packing problems), as **it allows to reduce the size of the problem**
- this entails a **model reformulation with decision variables in the form of paths (columns)**. Defining the RMP is generally straight-forward
- computing new promising feasible paths is done via the pricing problem. Defining its is also straight-forward, as they are inherited from the original “arc-based” formulation. **Defining the objective value and the proper decision variables is a bit harder and requires practice (and understanding of the dual values!)**
- a better choice for the initial set of paths can have an effect on the **convergence of the CG algorithm**

Some final words

- the CG algorithm is useful for those problems with many combinations between decision variables (e.g., routing and packing problems), as **it allows to reduce the size of the problem**
- this entails a **model reformulation with decision variables in the form of paths (columns)**. Defining the RMP is generally straight-forward
- computing new promising feasible paths is done via the pricing problem. Defining its is also straight-forward, as they are inherited from the original “arc-based” formulation. **Defining the objective value and the proper decision variables is a bit harder and requires practice (and understanding of the dual values!)**
- a better choice for the initial set of paths can have an effect on the **convergence of the CG algorithm**
- the CG algorithm can still be slow for large problems (there are acceleration techniques), and **is a heuristic. Hence, it is not guaranteed to converge to a global optimum**

AE4423-20: *Airline Planning and Optimization* - the Column Generation algorithm

Alessandro Bombelli
a.bombelli@tudelft.nl



Delft University of Technology, The Netherlands

December 5, 2024



Airline Planning and Optimisation - AE4423-20

Lecture 8 Crew Schedulling

Marta Ribeiro
Assistant Professor
Air Transport & Operation
Control & Operations

Delft University of Technology, The Netherlands

November, 2024

Course Information - Content

Management

- Planning structure
- Performance Indicators
- Demand and Supply
- Market share

Tactical Planning

- Passenger Mix Flow
- Fleet Assignment
- Aircraft Rotation
- Crew Scheduling**
- Maintenance



Strategical Planing

- Network Structure
- Demand Forecast
- Network Planning
- Fleet Planning

Operations Research

- MILP Models**
- Multi-commodity Flow Problems
- Shortest-path Algorithms**
- Column Generation Algorithm
- Dynamic Programming

Outline - Lecture 8

① Crew Scheduling Problem

- ② a Definitions
- ② b Considerations
- ② c Approach
- ② d Complexity

② Crew Pairing

- ③ a Problem Formulation

③ Crew Assignment

- ④ a Problem Formulation

④ Solution Approaches

- ④ a Column Generation Algorithm
- ④ b Multi-label Shortest Path Algorithm

Crew Scheduling Problem

An airline like KLM periodically schedules more than 500 flights per day to over 100 cities using more than 100 aircraft.

This might involve scheduling more than 2000 **cockpit crew** and 5000 **cabin crew** members on a monthly basis.

In general, **crew costs are the second largest operating expenses** (after fuel)

There is already a schedule with fleet types allocated to flights:

- Cockpit crew are heavily restricted to AC families
- Cabin crew are usually restricted to 2 or 3 AC families

Crew Scheduling Problem - Nomenclature

- **Duty period** - sequence of flight (and other duties) with short rest periods, called sits or idle times, separating them (typical duration - 1 day)
- **Brief** and **debrief** - periods at the beginning and end of a duty period
- **Pairing** - sequence of duty periods with overnight rest, called **layovers**, between them (variable duration, around 1 week)
- **Crew base** - station of origin of a crew, being the beginning and end of any pairing associated with the crew
- **Deadhead** - reposition of a crew, by flying them as passengers

Objective

Objective: minimize crew costs (or maximize preferences)

How: assigning crews to cover all flights for a given fleet type, while respecting

- Governing agencies regulations (e.g., FAA, US, EASA, EU)
- Labor agreements
- Airline requirements

Subject to:

- Space-time continuity
- Flight coverage
- Maximum number of crews
- Schedule balance constraints

Considerations

Domestic vs international operations:

- Domestic: H&S structure with connections - daily schedule
- International: sparse P2P structure - weekly schedule

Cabin crew composition may vary per flight:

- Crew salaries USA vs EU:
- USA: salary based on time flying + compensations
- EU: monthly payments

Cabin crew vs cockpit crew:

- Needed cabin crew amount can vary according to demand
- Cockpit crews for most airlines stay together (duty or pairing)
- Cockpit crew makeup is fixed
- Cabin crew composition may vary per flight

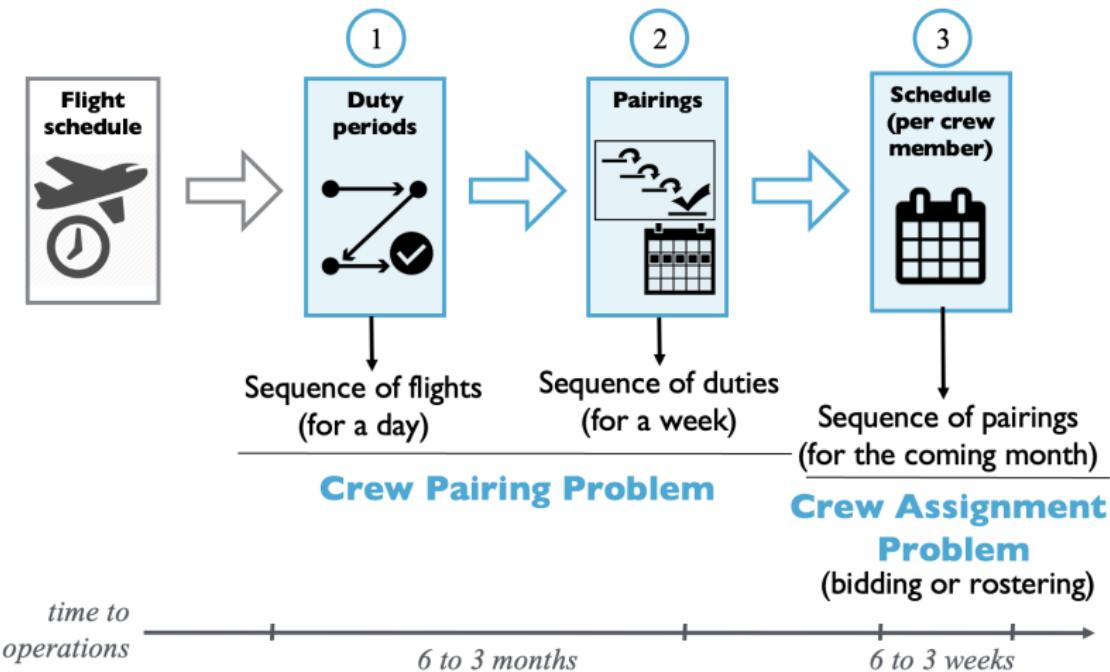
Complexity

The crew scheduling problem is considered to be one of the more challenging problems in the airline planning framework. **Why?**

- The **number of pairing is extremely large** for many airlines:
 - For a normal size airline, there could be more than 100 million legal pairings
 - For large airlines this value can rise above several billions
- Many **complex work rules** and safety regulations have to be satisfied:
 - E.g., the 8-in-24 rule (US) - extra rest if a pairing contains more than 8 hours of flying in any 24h periods (i.e., the layover between two days is extended, aka *compensatory rest*)
- Crew costs (specially compensation costs) depend on **complex crew pay guarantees and are highly nonlinear**

Approach

The crew scheduling problem is typically solved in three steps:



Approach (2)

Crew pairing and **crew assignment** problems are similar problems handled separately, in practice and in academic literature!

Let's look into these problems separately in the following sections.

Outline - Lecture 8

① Crew Scheduling Problem

- ② a Definitions
- ② b Considerations
- ② c Approach
- ② d Complexity

② Crew Pairing

- ③ a Problem Formulation

③ Crew Assignment

- ④ a Problem Formulation

④ Solution Approaches

- ④ a Column Generation Algorithm
- ④ b Multi-label Shortest Path Algorithm

Crew Pairing

The crew pairing problem starts with the definition of **duty periods**.

Duty rules:

- Day-long flights (or other duties) sequence in space and time
- Maximum flying time
- Minimum sit or idle times
- Maximum sit or idle times
- Maximum elapse time

Duty 'cost' is usually measured in minutes. It takes the maximum value of (in US):

- Total flying time
- Fraction of the elapse time (labor agreements)
- Minimum guaranteed duty pay (labor agreements)

Crew Pairing (2)

The crew pairing problem then consists on defining sequences of duties and layovers, that begin and end at a crew base.

Pairing rules:

- First duty starts/last duty ends at crew base
- Duties are sequential in space and time
- Maximum number of duties
- Minimum & maximum layover rest between duties
- Maximum number of days away from base

Pairing cost:

- Maximum value of (US):
 - Total cost associated to the flying time (duty cost)
 - Cost obtained via a fraction of the elapse time ($fp \times TAFB$)
 - Minimum guaranteed duty pay (min guarantee pay)
- Plus, costs associated with layover between duties (e.g., meals, lodging)

Crew Pairing (3)

The crew pairing problem is typically solved in two (EU) or three (US) stages:

- **Weekly** - considers the set of flights for an entire week

or (in the US):

- **Daily**: considers the set of frequent flights which are flown at least four days per week
- **Weekly exceptions**: constructs new pairings that include flights not flown on certain days of the week and flights that are flown 3 or fewer days per week, in order to guarantee time and space pairing continuity

plus:

- **Transition**: considers the period of transition between schedule seasons (e.g., every quarter or month) in order to guarantee time and space pairing continuity

Crew Pairing (3)

Example of a list of pairing for multiple days:

	MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT
PA_0001	PA_0001	ME_003	ME_003	ME_003	ME_003	RP	RP						
PA_0002	PA_0002	ME_002	ME_002	RP	RP	RP							
PA_0003	PA_0003	NA_005	NA_005	NA_005	RP	RP	RP						
PA_0004	PA_0004	CA_004	CA_004	RP	RP	RP	RP						
PA_0005	PA_0005	ME_001	ME_001	RP	RP	RP							
PA_0006	PA_0006	CA_001	CA_001	CA_001	RP	RP	RP						
PA_0007	PA_0007	NA_004	NA_004	NA_004	RP	RP	RP						
PA_0008	PA_0008	AF_003	AF_003	RP	RP								
PA_0009	PA_0009	NA_002	NA_002	NA_002	NA_002	RP	RP						
PA_0010	PA_0010	AF_001	AF_001	AF_001	RP	RP	RP						
PA_0011		PA_0011	ME_002	ME_002	RP	RP	RP						
PA_0012		PA_0012	CA_003	CA_003	RP	RP	RP	RP					
PA_0013		PA_0013	ME_003	ME_003	ME_003	ME_003	ME_003	RP	RP	RP			
PA_0014		PA_0014	CA_004	CA_004	RP	RP	RP	RP					
PA_0015		PA_0015	ME_001	ME_001	RP	RP	RP						
PA_0016		PA_0016	CA_001	CA_001	RP	RP	RP						
PA_0017		PA_0017	AF_002	AF_002	AF_002	AF_002	AF_002	AF_002	RP	RP	RP	RP	RP
PA_0018		PA_0018	AF_003	AF_003	RP	RP							
PA_0019		PA_0019	AF_001	AF_001	AF_001	RP	RP	RP					
PA_0020		PA_0020	NA_005	NA_005	NA_005	NA_005	NA_005	RP	RP	RP	RP		
PA_0021		PA_0021	ME_001	ME_001	RP	RP	RP	RP					
PA_0022		PA_0022	NA_004	NA_004	NA_004	NA_004	RP	RP	RP	RP			

RP - Rest Period

Crew Pairing - Problem Formulation

Goal: Given a set of flights, choose a minimum cost set of pairings such that every flight is covered exactly once (i.e. every flight is contained in one and only one pairing)

Sets:

P : Set of feasible pairings

F : Set of daily flights

Parameters:

c_p : Cost associated with pairing

δ_f^p : equal to 1 if flight f is part of pairing p ; equal to 0 otherwise

Decision Variables:

x_p : 1 if pairing p is chosen, and 0 otherwise

Crew Pairing - Problem Formulation (2)

Objective Function:

$$\min \sum_{p \in P} c_p \times x_p \quad (\text{OF})$$

Subject to:

$$\sum_{p \in P} \delta_f^p \times x_p = 1, \quad \forall f \in F \quad (\text{C1}) \text{ Every single flight is covered (only) once}$$

$$x_p \in \{0, 1\}, \quad \forall p \in P \quad (\text{C2})$$

where the cost of each pairing can result, e.g., from:

$$c_p = \max \left\{ \sum_{d \in P} \text{duty cost of } d, f_p \times \text{TAFB}, \min \text{ guarantee pay} \right\} + \text{layover costs}$$

Note: TAFB = Time Away From Base

Model Size ($|F^k| = |\text{daily flights assigned to } k|$; $|P^k| = |\text{feasible pairings for } k|$):

- Constraints: $|F^k|$
- Variables: $|P^k|$

Crew Pairing - Constraints

Balancing constraints:

- Many airlines also add **crew base balancing constraints**
- To ensure a good distribution of work over the set of crew bases, for instance:
 - Number of hours of work contained in the chosen pairings starting at a given crew base. Must be between specific lower and upper bounds, defined as a function of the number of crews stationed at that crew base

Is this an easy problem?

- Linear objective function
- No complex feasibility rules
- Easy to write/intuitive
- Small number of constraints

Outline - Lecture 8

① Crew Scheduling Problem

- ② a Definitions
- ② b Considerations
- ② c Approach
- ② d Complexity

② Crew Pairing

- ③ a Problem Formulation
- ③ b Constraints

③ Crew Assignment

- ④ a Problem Formulation
- ④ b Constraints

④ Solution Approaches

- ⑤ a Column Generation Algorithm
- ⑤ b Multi-label Shortest Path Algorithm

Crew Assignment

Monthly problem of allocating crews to cover all pairings selected for all weeks in the month. The result is a sequence of pairings per each crew member, called **schedules** or **rosters**

Schedules rules (in a month):

- Maximum flying time (block-hours)
- Minimum number of consecutive days off
- Minimum number of days off (total)
- Maximum duty time
- Minimum rest time between pairings

Schedule optimization is associated with:

- Crew preferences/satisfaction
- Workload balance
- (In theory, minimization of costs have already been addressed in the crew pairing problem)

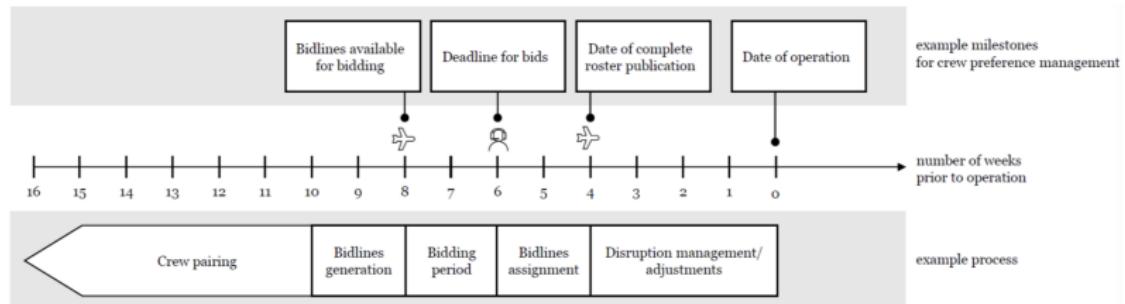
Crew Assignment (2)

Differences between US and EU:

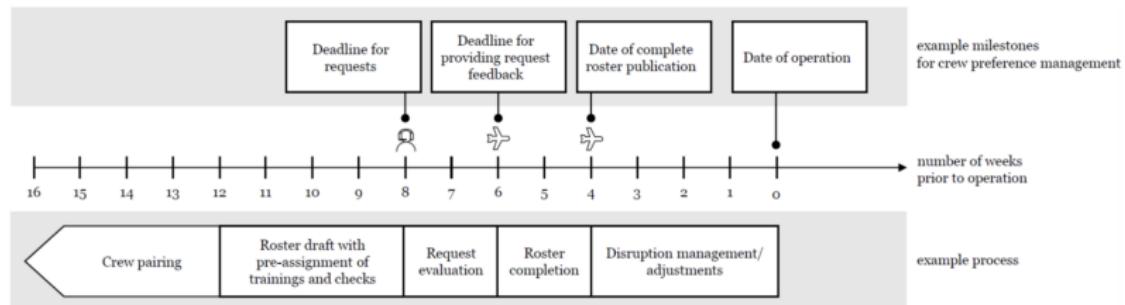
- US - two stages bidline process:
 - ① 1st stage - construction of schedules (set of lines of pairing)
 - ② 2nd stage - crew member bid for their preferred line (often seniority rule is used)
- EU - one stage rostering process:
 - ① Schedules are constructed (rosters) and assigned to crew members
 - ② Individual requests, such as vacation time, training time, etc., are considered when selecting feasible schedules per crew member

Crew Assignment (3)

Crew assignment according to bidlines:



Crew rostering, with preferences:



👤 : responsibility of crew members

✈️ : responsibility of airline

Crew Assignment (4)

Example of a crew roster for multiple crew members (one per row)

Note: NA - a previous pairing was assigned in the past

	MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE
L_001	NA	NA	NA	NA	NA	PA_0059	CA_004	CA_004	RP	RP	RP	RP				
L_002	NA	PA_0011	ME_002	ME_002	RP	RP	RP									
L_003	NA	NA	NA	PA_0034	CA_002	CA_002	CA_002	RP	RP	RP						
L_004	PA_0009	NA_002	NA_002	NA_002	NA_002	RP	RP									
L_005	NA	NA	NA	NA	PA_0048	CA_001	CA_001	CA_001	RP	RP	RP					
L_006	NA	NA	NA	NA	NA	PA_0054	NA_005	NA_005	NA_005	RP	RP	RP				
L_007	NA	NA			PA_0045	AF_002	AF_002	AF_002	AF_002	AF_002	AF_002	AF_002	RP	RP	RP	
L_008	NA	NA	NA	NA	NA	PA_0068	AF_004	AF_004	AF_004	AF_004	AF_004	AF_004	RP	RP	RP	
L_009		PA_0025	ME_002	ME_002	RP	RP	RP									
L_010	PA_0010	AF_001	AF_001	AF_001	RP	RP	RP									
L_011	NA	NA	NA		PA_0065	NA_003	NA_003	NA_003	NA_003	NA_003	NA_003	NA_003	RP	RP	RP	RP
L_012	NA	NA	NA	NA	PA_0047	ME_002	ME_002	ME_002	RP	RP	RP					
L_013	NA	NA	NA	NA	NA	PA_0051	ME_002	ME_002	ME_002	RP	RP	RP	RP	RP	RP	
L_014	NA	NA	NA	PA_0037	NA_002	NA_002	NA_002	NA_002	RP	RP	RP					
L_015	NA	NA	NA	PA_0046	AF_001	AF_001	AF_001	AF_001	RP	RP	RP	RP				
L_016	NA	NA	NA	NA												
L_017	NA	NA	PA_0026	CA_004	CA_004	RP	RP	RP	RP							
L_018	NA	NA	PA_0038	ME_003	ME_003	ME_003	ME_003	ME_003	RP	RP	RP					
L_019		PA_0023	AF_003	AF_003	RP	RP										
L_020	NA	NA	NA	NA	NA	PA_0070	NA_001	NA_001	NA_001	RP	RP	RP				
L_021	NA	NA	NA	NA	NA											
L_022	NA	NA	NA	NA	PA_0050	NA_004	NA_004	NA_004	RP	RP	RP					
L_023	NA	NA	NA	NA	NA	NA	PA_0063	AF_003	AF_003	RP	RP					
L_024		PA_0020	NA_005	NA_005	NA_005	NA_005	NA_005	NA_005	RP	RP	RP					
L_025	PA_0013	ME_003	ME_003	ME_003	ME_003	ME_003	ME_003	ME_003	RP	RP	RP					
L_026	NA	NA	NA	NA	NA	PA_0056	NA_002	NA_002	NA_002	RP	RP	RP				
L_027	PA_0004	CA_004	CA_004	RP	RP	RP	RP									
L_028	NA	NA	PA_0029	CA_003	CA_003	RP	RP	RP	RP							
L_029	NA	NA	NA	NA	NA	PA_0052	AF_001	AF_001	AF_001	RP	RP	RP				
L_030	NA	NA	NA	NA	NA	NA										

Crew Assignment - Problem Formulation

The crew assignment problem consists on solving separate monthly rostering problems for each crew member.

Crew types defined according to:

- Rank (e.g., Captain, First officer, Flight engineer, senior cabin crew, grade one)
- Fleet family (e.g., Airbus 320, Boeing 777)

For a given crew, the **model inputs** include:

- Set of pairings that must start at each day
- Number of crew members of each type that can be assigned

Constraints:

- Each pairing in the time horizon has to be covered by a group of crew members, as specified by the underlying fleet,
- Each crew member needs to be assigned to exactly one work schedule (if the airline has too many crew members, some can be allocated to an *empty* or *null* schedule)

Crew Assignment - Problem Formulation (2)

Goal: Given a set of schedules, allocate rosters to the crew in order to maximise the individual crew preferences or bids.

Sets:

M : Set of crew members of a given type

S^m : Set of work rosters that are feasible for crew member m

P : Set of (dated) pairings to be covered by the crew type considered

Parameters:

n_p : Min number of crew members that need to be assigned to pairing p

c_s^m : cost of assigning employee m to roster s

γ_p^s : equal to 1 if pairing p is part of schedule s ; equal to 0 otherwise

Decision Variables:

x_s^m : 1 if crew member m is assigned to roster s , and 0 otherwise

Crew Assignment - Problem Formulation (3)

Objective Function:

$$\min \sum_{m \in M} \sum_{s \in S^m} c_s^m \times x_s^m \quad (\text{OF})$$

Subject to:

$$\sum_{m \in M} \sum_{s \in S^m} \gamma_p^s \times x_s^m \geq n_p, \quad \forall p \in P \quad (\text{C1}) \text{ Number of crew members per pairing}$$

$$\sum_{s \in S^m} x_s^m = 1, \quad \forall m \in M \quad (\text{C2}) \text{ Each crew member in a rooster}$$

$$x_s^m \in \{0, 1\}, \quad \forall s \in S^m, m \in M \quad (\text{C3})$$

Model Size ($|S^m| = |\text{possible schedules to } m|$; $|P| = |\text{pairings to cover}|$; $|M| = |\text{crew members}|$):

- Constraints: $|P| + |M|$
- Variables: $\sum_m |S^m|$

Outline - Lecture 8

① Crew Scheduling Problem

- ② a Definitions
- ② b Considerations
- ② c Complexity
- ② d Approach

② Crew Pairing

- ③ a Problem Formulation

③ Crew Assignment

- ④ a Problem Formulation

④ Solution Approaches

- ⑤ a Column Generation Algorithm
- ⑤ b Multi-label Shortest Path Algorithm

Column Generation Algorithm

Given the complexity of the problem, traditional approaches involve:

- ① Use the previous set of pairings, adapted by hand some of them and add a few more pairings
- ② Use heuristics to improve current solution
- ③ Solve an IP problem only considering a subset of the possible pairings



Column Generation Algorithm

Column Generation Algorithm - Problem Formulation

Crew Pairing Problem - Objective Function:

$$\min \sum_{p \in P} c_p \times x_p \quad (\text{OF})$$

Subject to:

$$\sum_{p \in P} \delta_f^p \times x_p = 1, \quad \forall f \in F \quad (\text{C1}) \longrightarrow \pi_f$$

- ① Relax the integrity property of the problem: $x_p \in \mathbb{Z}^+, \forall p \in P$
- ② Associated π_f as the dual variables of constraints (C1)
- ③ Compute slackness for each pairing p

$$slack = \sum_{i \in N} a_{ij} \times y_i - c_f \quad \implies$$

$$slack = \sum_{f \in F} \delta_f^p \times \pi_f - c_p$$

- ④ If slackness is positive >> add column (*pairing p)

Column Generation Algorithm (2)

How to generate new pairings in our relaxed version of the problem?

There could be thousands or billions of possible pairings – enumeration is computational unrealistic for most cases

Start with an initial set of columns for feasibility:

- Previously used pairings
- Simple, but expensive, pairings (e.g., 1 flight = 1 pairing)
- Add slack variables to constraints (C1) and penalize these variables in the objective function (OF)

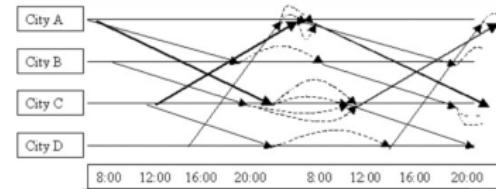
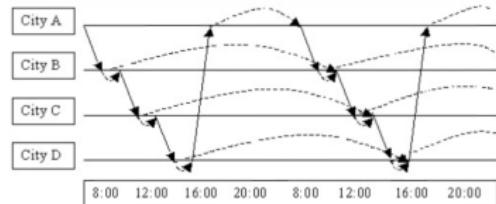
Generation of new columns:

- Heuristics to explore a limited sub-set of pairings:
 - The pairings can be randomly generated or generated based on a limited number of crew connections (or using historical data on useful connections)
- Use a shortest-path in a network of duties in a time-space network

Column Generation Algorithm (3)

Network representations:

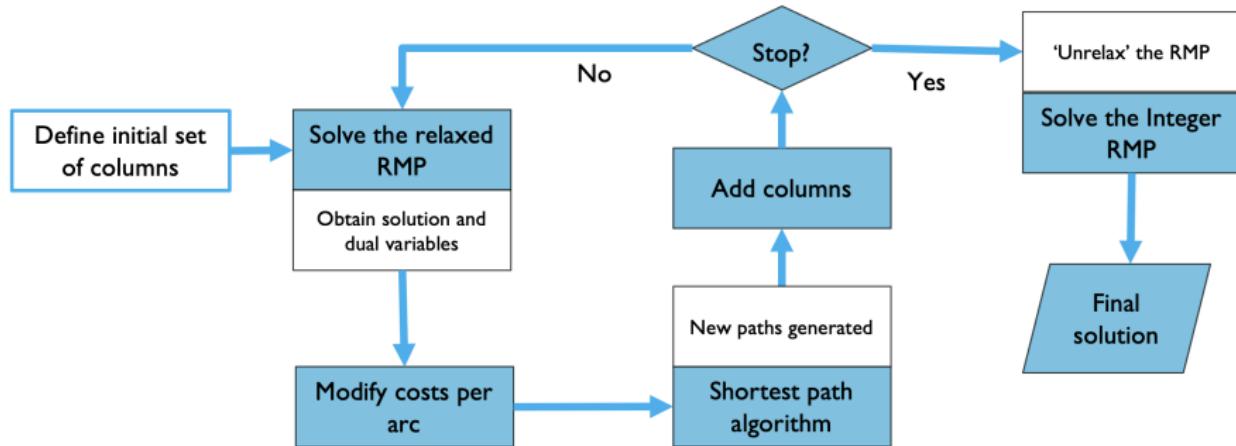
- Option 1: Flight network:
 - There is an arc for each flight in the schedule
 - Connections arcs exist for nodes at the same airport and the time between them is enough for a sit or layover times
- Option 2: Duty period network:
 - Built already by considering duty period rules into the network (e.g., the sit and layover times)
 - Results in larger arc set as there could be multiple duty periods associated with the same flight



Note: in both networks, a **source node** is connected to the departure node of each flight that originates at a specific crew base; and a **sink node** to the arrival node of every flight ending at that crew base.

Source: Barnhart et al (2003)

Column Generation Algorithm (4)



Modified cost per arc:

- Flight network:

$$c_f' = c_f + \pi_f$$

- Duty period network:

$$c_{duty}' = c_{duty} + \sum_{f \in F^P} \delta_f^P \times \pi_f$$

Multi-label Shortest Path Algorithm

The **Multi-label shortest path** algorithm consider the case of **finding a minimum cost path** while other metrics are involved in the decision or are limited in terms of feasible values.

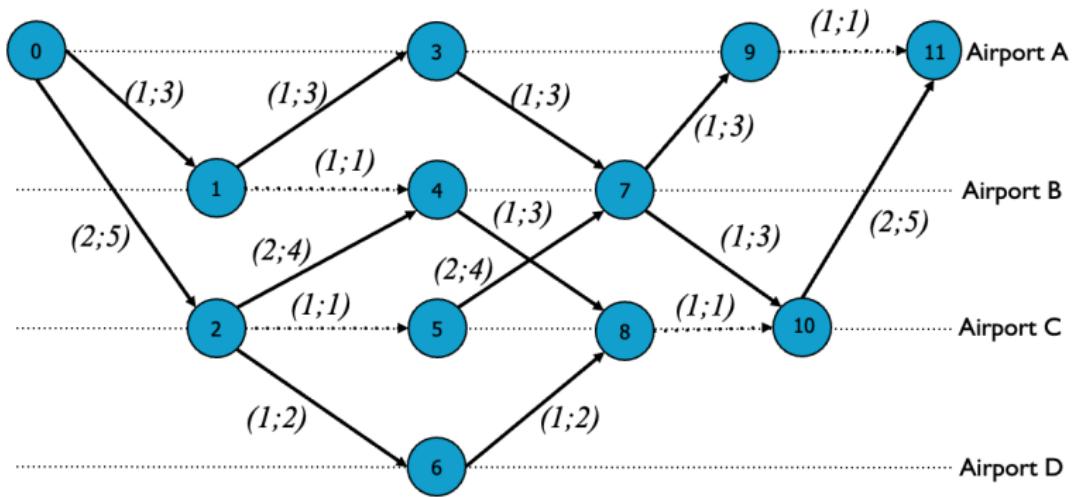
Example of a 5 days pairing problem:

- Assume the following label set:
 - Label $c_p(j)$ be the cost associated to a path p to node j
 - Label $t_p(j)$ be the flying time on a path p to node j
 - Label $d_p(j)$ be the number of duties on a path p to node j
- The goal is to obtain the least cost given that:
 - No more than 3 duties are allowed per path (i.e., pairing)
 - The total flying time should not be longer than 15 hours

Multi-label Shortest Path Algorithm (2)

- Only paths with labels within the limits are considered (the rest are excluded)
- Any path p' into j that is dominated by another path p at node j should be excluded from the analysis.
 - A path p' is dominated by path p at node j iff all the labels are worse for p' :
$$c'_p(j) > c_p(j), \quad t'_p(j) > t_p(j) \quad \text{and} \quad d'_p(j) > d_p(j)$$
 - In this case, we will eliminate p' from our analysis
- If p' not dominated, we will keep both p' and p in our analysis. The label information for both are stored and we continue exploring both paths for the following nodes in our network.

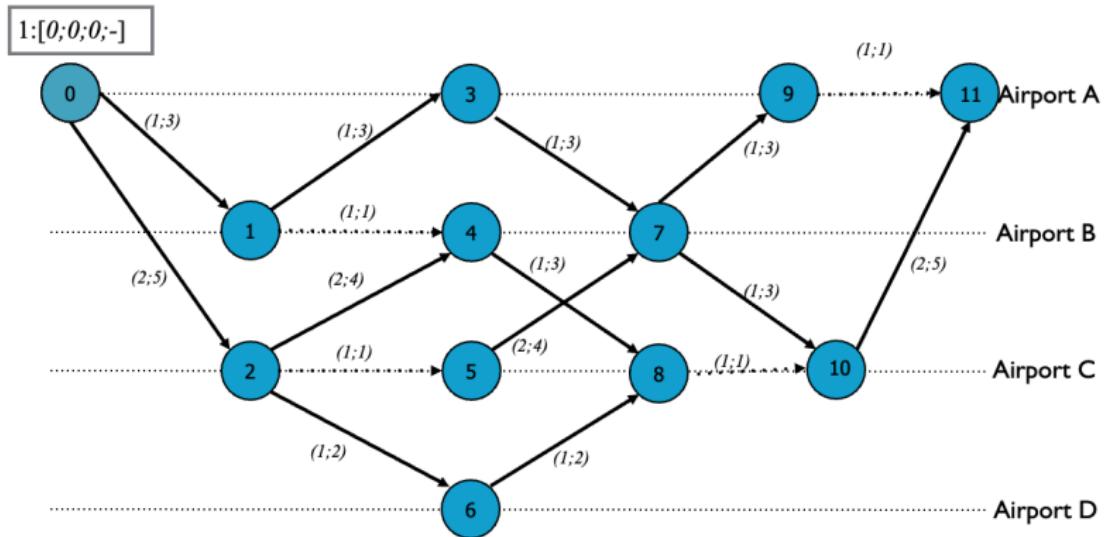
Multi-label Shortest Path Algorithm (3)



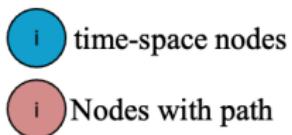
At each arc:
 $(cost; time)$

i time-space nodes
i Nodes with path

Multi-label Shortest Path Algorithm (4)



At each arc:
 $(cost; time)$



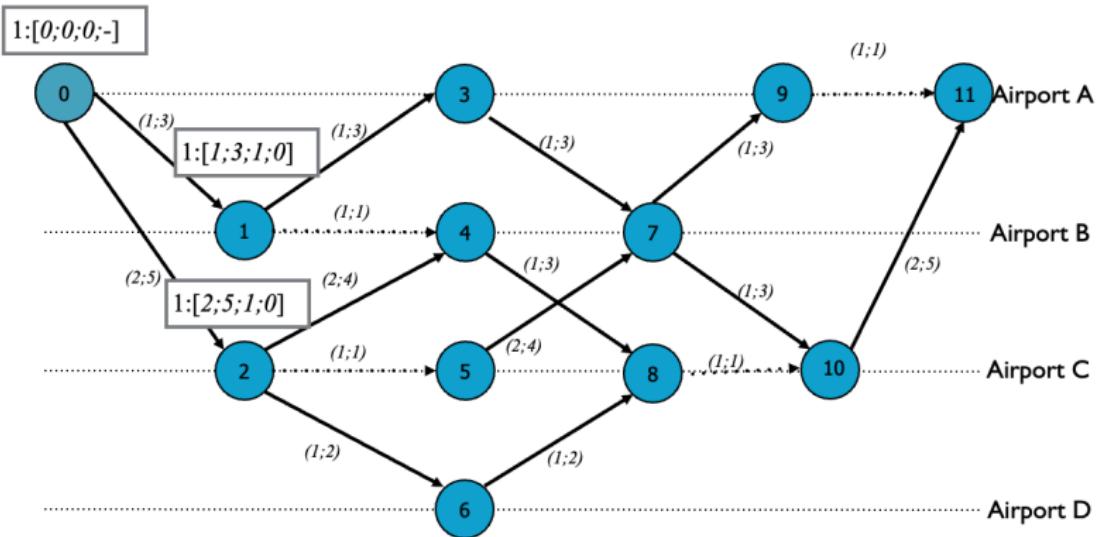
~~p:[c;t;d;pr]~~ path dominated

p:[c;t;d;pr] path unfeasible

At each node:

path: $[cost; time; \#duties, prec]$

Multi-label Shortest Path Algorithm (5)



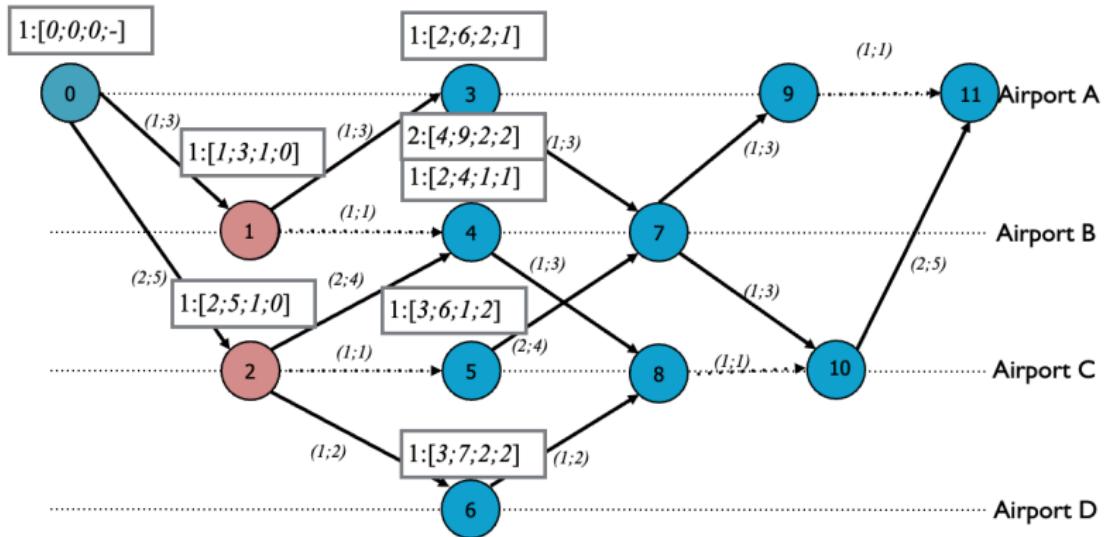
At each arc:
 $(cost; time)$

At each node:
path: [cost; time; #duties, prec]

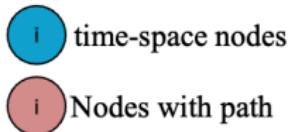
- | time-space nodes
- | Nodes with path

$p:[c:t;d,pr]$	path dominated
$p:[c:t;d,pr]$	path unfeasible

Multi-label Shortest Path Algorithm (6)



At each arc:
 $(cost; time)$



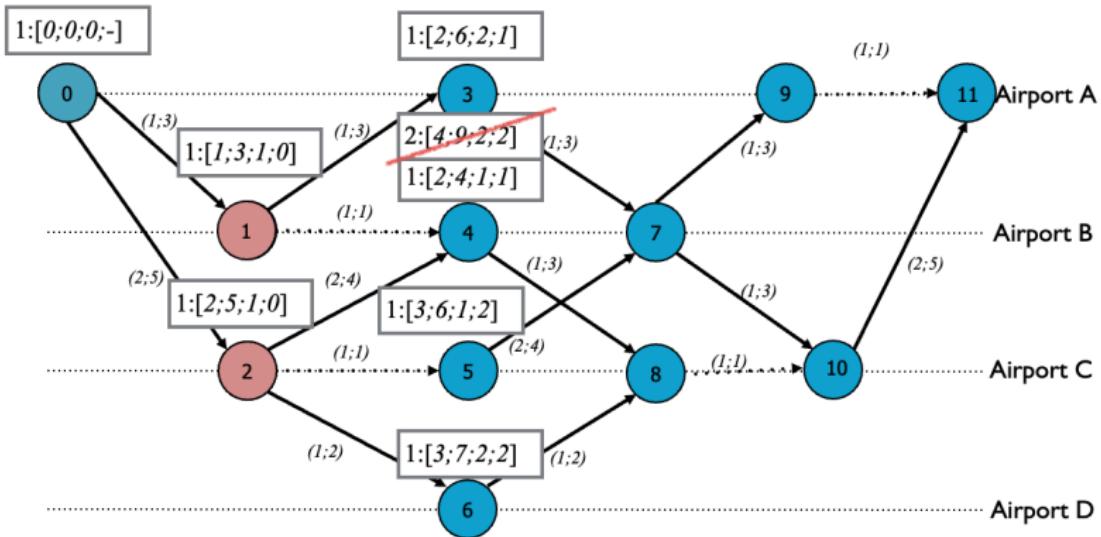
~~p:[c;t;d,pr]~~ path dominated

p:[c;t;d,pr] path unfeasible

At each node:

path: $[cost; time; \#duties, prec]$

Multi-label Shortest Path Algorithm (7)



At each arc:
 $(cost; time)$

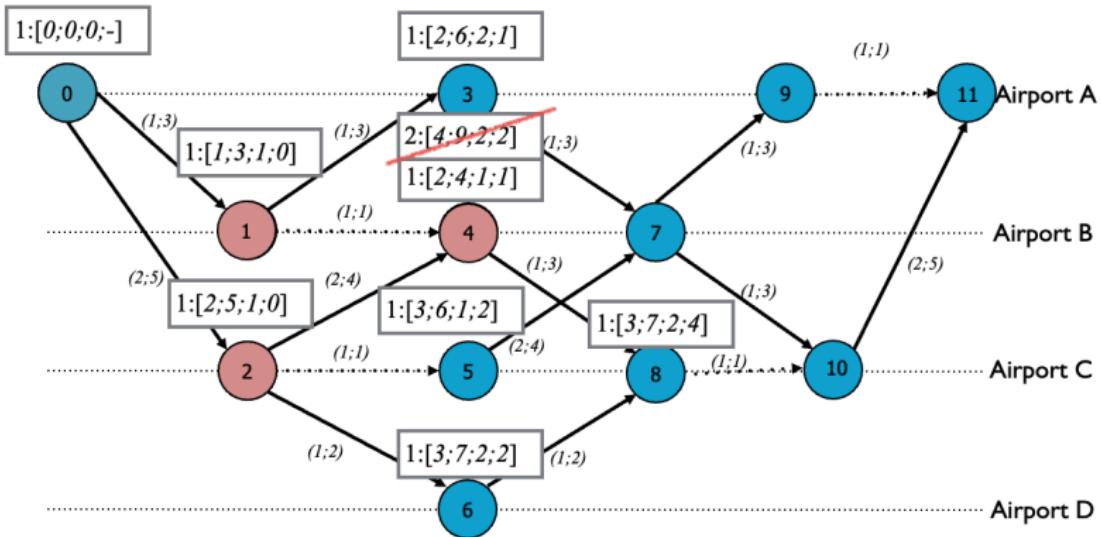
i time-space nodes
 i Nodes with path

$p:[c;t;d,pr]$ path dominated
 $p:[c;t;d,pr]$ path unfeasible

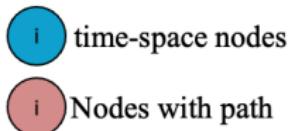
At each node:

path: $[cost; time; \#duties, prec]$

Multi-label Shortest Path Algorithm (8)



At each arc:
 $(cost; time)$

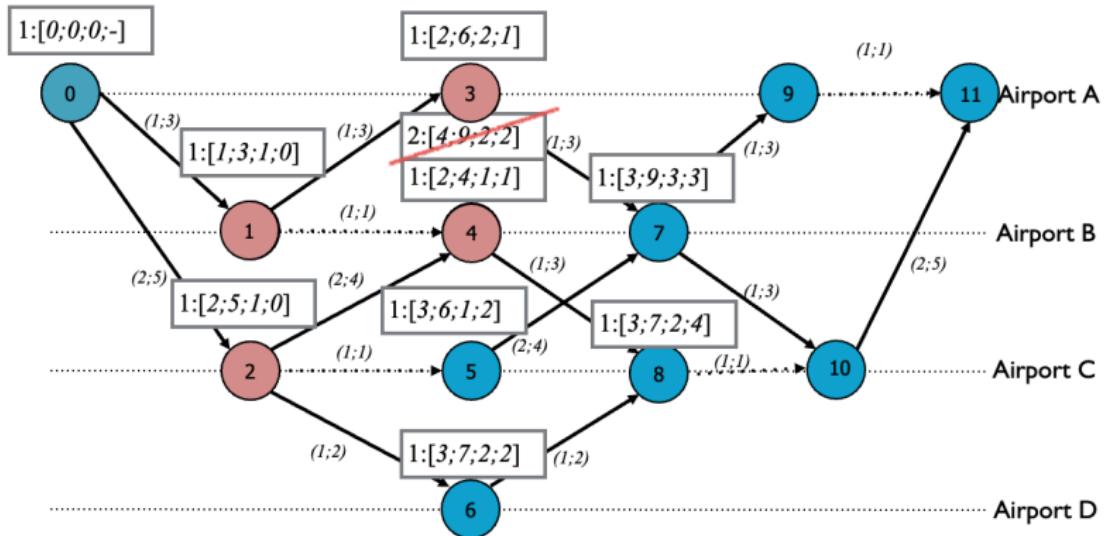


~~p:[c;t;d;pr]~~ path dominated
~~p:[c;t;d;pr]~~ path unfeasible

At each node:

path: $[cost; time; \#duties, prec]$

Multi-label Shortest Path Algorithm (9)



At each arc:
 $(cost; time)$

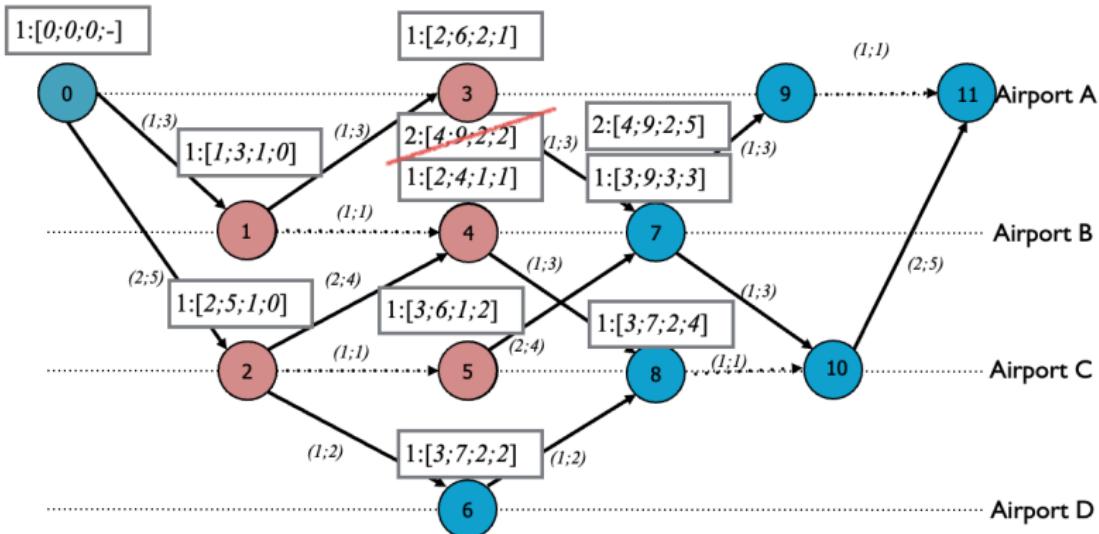
i time-space nodes
 i Nodes with path

$p:[c;t;d,pr]$ path dominated
 $\cancel{p:[c;t;d,pr]}$ path unfeasible

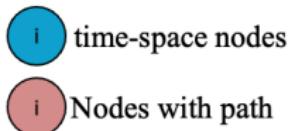
At each node:

path: $[cost; time; \#duties, prec]$

Multi-label Shortest Path Algorithm (10)



At each arc:
 $(cost; time)$



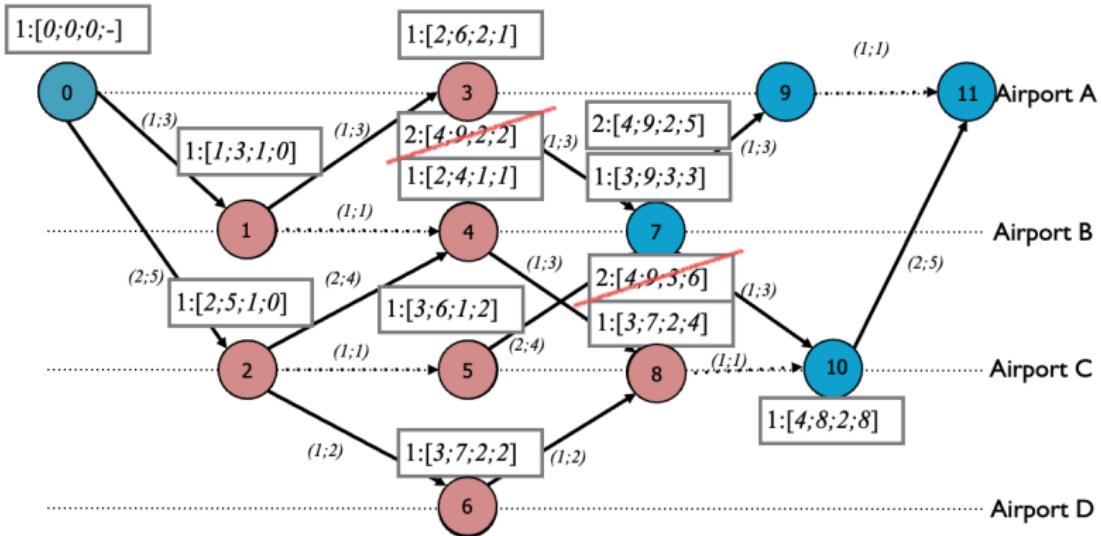
~~p:[c;t;d,pr]~~ path dominated

~~p:[c;t;d,pr]~~ path unfeasible

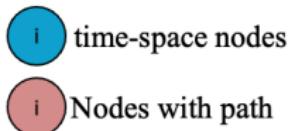
At each node:

path: $[cost; time; \#duties, prec]$

Multi-label Shortest Path Algorithm (11)



At each arc:
 $(cost; time)$



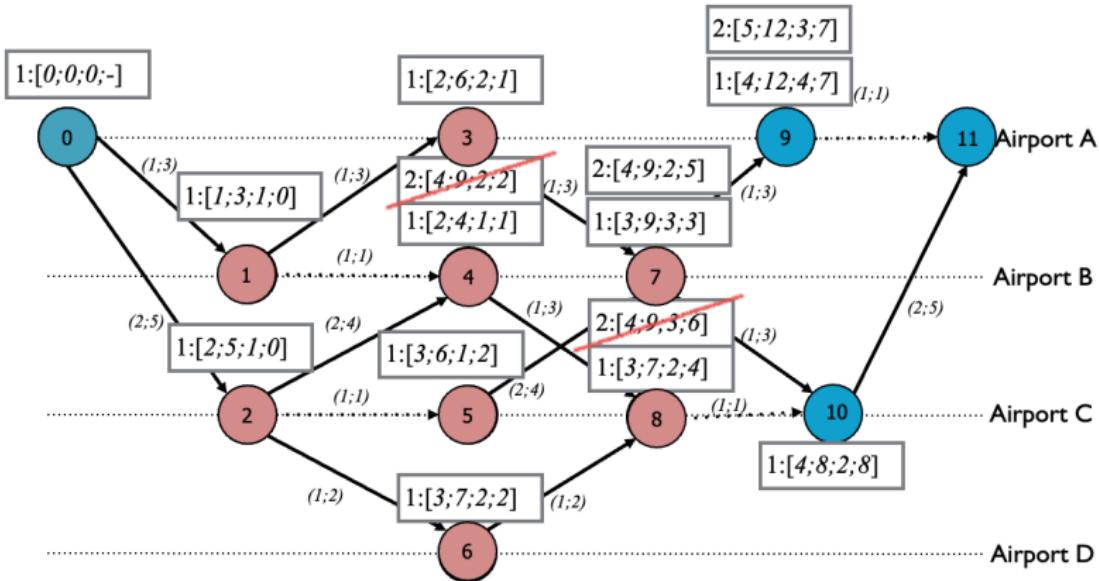
~~p:[c;t;d;pr]~~ path dominated

At each node:

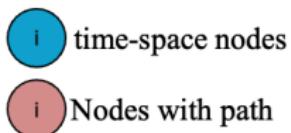
path: $[cost; time; \#duties, prec]$

~~p:[c;t;d;pr]~~ path unfeasible

Multi-label Shortest Path Algorithm (12)

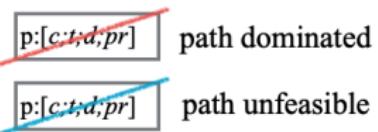


At each arc:
 $(cost; time)$

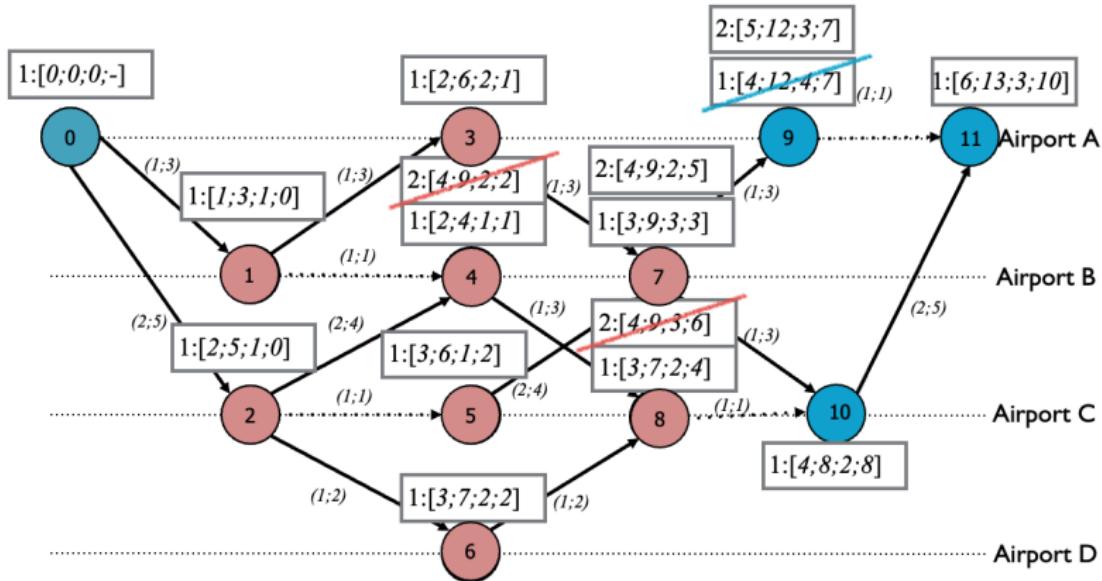


At each node:

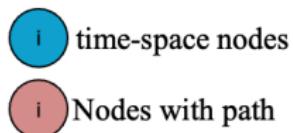
path: $[cost; time; \#duties, prec]$



Multi-label Shortest Path Algorithm (13)



At each arc:
(cost, time)

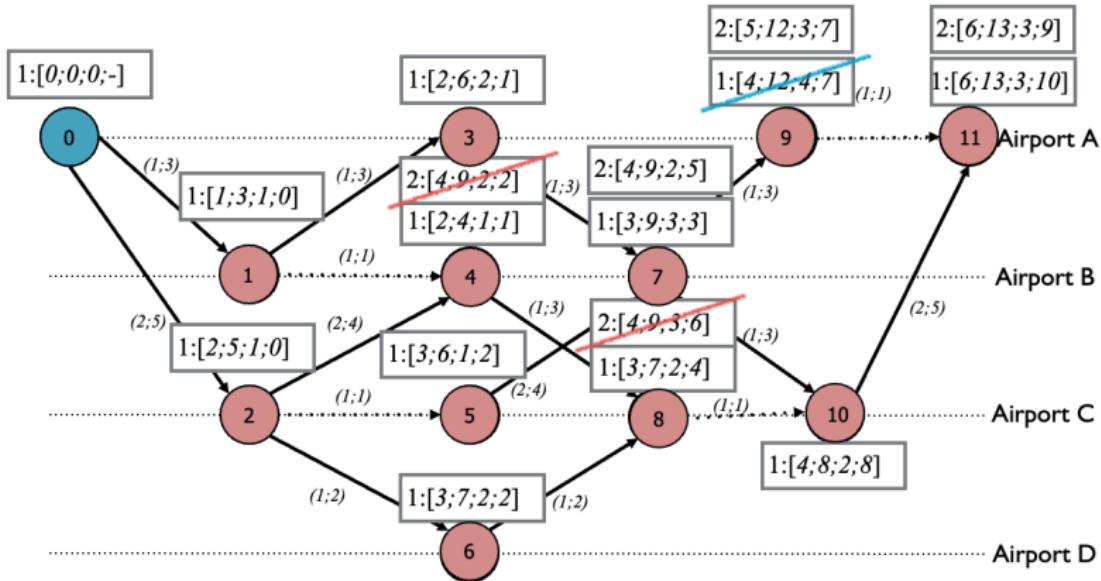


~~p:[c;t;d;pr]~~ path dominated
~~p:[c;t;d;pr]~~ path unfeasible

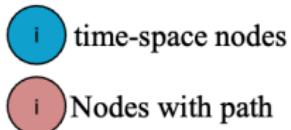
At each node:

path: [cost; time; #duties, prec]

Multi-label Shortest Path Algorithm (14)



At each arc:
 $(cost; time)$

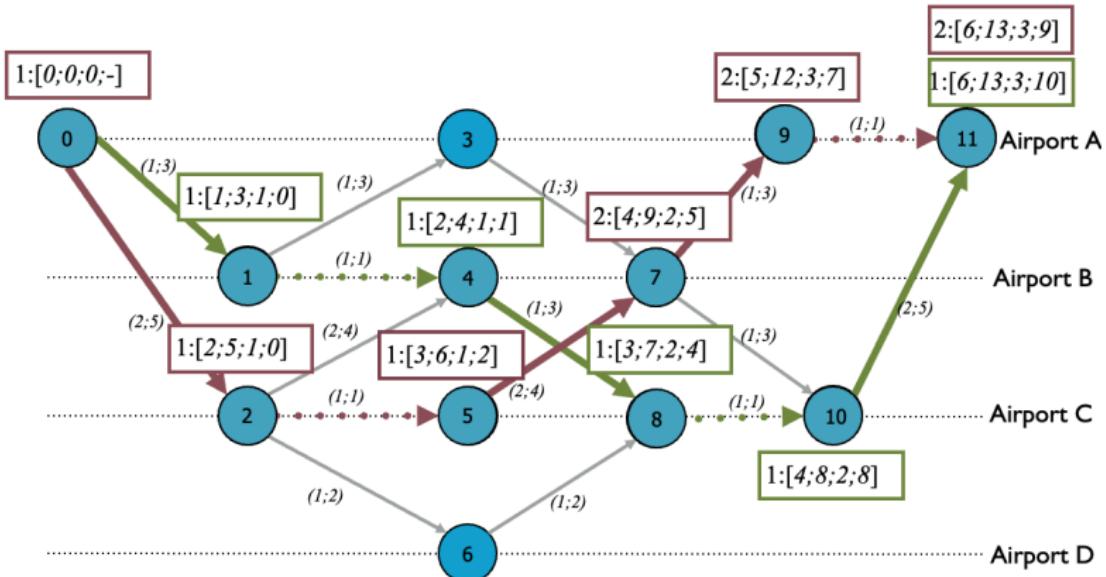


~~$p:[c:t;d,pr]$~~ path dominated
 $\cancel{p:[c:t;d,pr]}$ path unfeasible

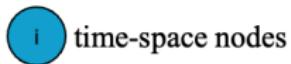
At each node:

path: $[cost; time; \#duties, prec]$

Multi-label Shortest Path Algorithm (15)

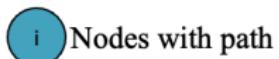


At each arc:
(cost; time)



path 1

At each node:



path 2

path: [cost; time; #duties, prec]

Airline Planning and Optimisation - AE4423-20

Lecture 9

Maintenance Schedulling

Marta Ribeiro
Assistant Professor
Air Transport & Operations
Control & Operations

Delft University of Technology, The Netherlands

November, 2024

Course Information - Content

Management

- Planning structure
- Performance Indicators
- Demand and Supply
- Market share

Tactical Planning

- Passenger Mix Flow
- Fleet Assignment
- Aircraft Rotation
- Crew Scheduling
- Maintenance



Strategical Planing

- Network Structure
- Demand Forecast
- Network Planning
- Fleet Planning

Operations Research

MILP Models

- Multi-commodity Flow Problems
- Shortest-path Algorithms
- Column Generation Algorithm

Dynamic Programming

Outline - Lecture 9

① Maintenance

- ② a Hangar Checks
- ② b Line Maintenance
- ② c Maintenance Costs

② Maintenance Scheduling Problem

- ③ a Maintenance Scheduling Model
- ③ b Other Requirements

③ Dynamic Programming

- ④ a Challenges
- ④ b A Real-Life Example

Maintenance

Aircraft need to be **regularly maintained** to ensure compliance with **safety** and continuous **airworthiness requirements**.

- Aircraft maintenance involves overhaul, repair, inspection or modification of an aircraft and aircraft components
- It can be performed in:
 - Routine or hangar checks - e.g., A, C, D checks
 - Line maintenance (i.e., during TAT at airports or overnight)



Hangar Checks

Check	Frequency	Duration	Notes
A	Every 2 to 3 months (400-600 flight hours or 200–300 cycles)	± 24 hours	Regular tasks, that vary from an A-check to the other
B	Every 6 to 8 months	± 48 hours	Most of the times, it is merged with A-checks
C	Every 2 to 3 years	1 to 2 weeks	Large majority of the aircraft components are inspected
Intermediate Layover (IL)*	Every 3 to 5 years	2 to 4 weeks	Numerous items, systems and equipment, are tested. Cabin components overhauled.
D	Every 6 to 10 years	4 to 6 weeks	The entire structure is inspected. Cabin interior, engines, landing gears are dismantled.

*Only employed by some airlines for certain aircraft

Hangar Checks (2)

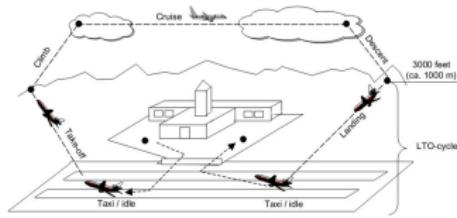
Aircraft is aged by:



Calendar day DY



Flight time FH



Flight cycle FC

Maintenance is pre-scheduled according to intervals:

0 DY	120 DY	0 DY
0 FH	750 FH	0 FH
0 FC	750 FC	0 FC

120 DY	0 DY
750 FH	0 FH
750 FC	0 FC

A-Check:

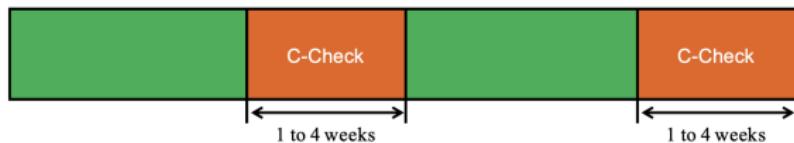


0 DY	730 DY
0 FH	7500 FH
0 FC	5000 FC

0 DY
0 FH
0 FC

730 DY
7500 FH
5000 FC

C-Check:



Line Maintenance

Aircraft maintenance activity carried out whilst the aircraft remains in the operating environment.



Typical **line maintenance** items are:

- General inspection before readying an aircraft for flight
- Daily check - performed every 24 to 48 hours
- Weekly check - performed every 7 to 8 days
- Any unscheduled maintenance resulting from unforeseen events
- Scheduled checks that do not require specialised training, equipment, or facilities.

Maintenance Costs

From an IATA study involving 51 airlines:

Maintenance Costs	Average [US\$/Year]	Min [US\$/Year]	Max [US\$/Year]
Per Airline	295M	0.95M	2.28B
Per Aircraft	3.6M	0.67M	9.3M
Per Flight Hour	1087	287	2841
Per Flight Cycle	2681	465	11937

For an A320 type aircraft (estimations):

Check	Labour-hours	Material Cost [US\$]	Total Cost [with Labour Cost=60US\$/h]
Daily	4	500	740
Weekly	10	700	1300
A	80	5500	10300
C	2 000	28500	148500
IL	14 500	0.38M	1.25M
D	20 000	1.5M	2.7M

Source: IATA's Maintenance Cost Task Force. Airline Maintenance Cost Executive Commentary – An Exclusive Benchmark Analysis, 2015.

Maintenance Costs (2)

The costs increase if we consider the loss of revenue given the unavailability of the aircraft.

- For 24 hours unavailable (estimations):

Aircraft	Seats	Revenue Losses
B777 / A350	310 - 370	± 0.2M US\$
B787 / A330	240 - 290	± 0.15M US\$
B737 / A319-20	120 - 140	± 75 000 US\$
ERJ190 / CS100	80 - 110	± 60 000 US\$

Outline - Lecture 9

① Maintenance

- ② a Hangar Checks
- ② b Line Maintenance
- ② c Maintenance Costs

② Maintenance Scheduling Problem

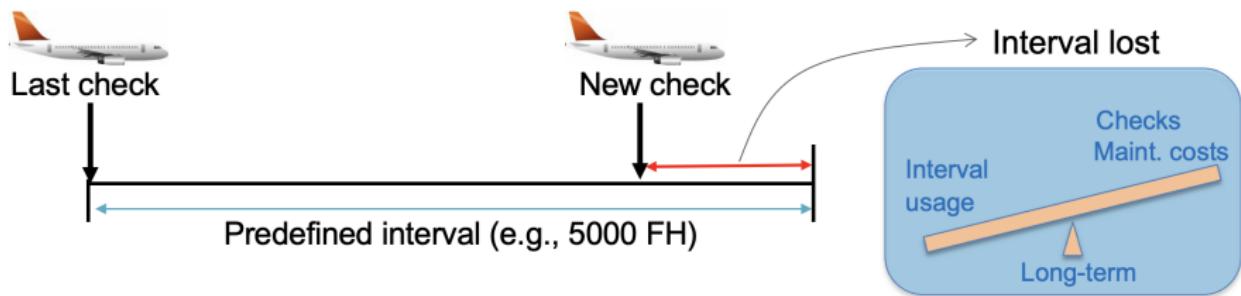
- ③ a Maintenance Scheduling Model
- ③ b Other Requirements

③ Dynamic Programming

- ④ a Challenges
- ④ b A Real-Life Example

Maintenance Scheduling Problem

The problem: To define the starting (and end) date of a set of maintenance checks that minimize the check interval lost



Subject to:

- The given a set of aircraft in a fleet
- The amount of resources available for maintenance (e.g., hangar slots, maintenance teams) every day
- The list of predefined maintenance checks to scheduled, associated with a given duration and due date

Maintenance Scheduling Model

Objective Function:

- Minimise maintenance costs
- Minimise the interval lost (\times costs) (OF)
- Maximise aircraft availability (for operations)

Subject to:

- Maintenance before interval deadline (C1)
- Maintenance elapse time - to track the resources utilisation (C2)
- Maximum utilisation of maintenance slots (C3)
- Maximum workload (C4)

Maintenance Scheduling Model (2)

Sets:

K : Set of aircraft

J : Set of maintenance checks

T : Time periods

Parameters:

$D_{k,j}$: Deadline to start maintenance check j on aircraft k

α_j : Costs of anticipating maintenance check j one day (i.e., lost from
not using the full interval)

$ET_{k,r}$: Elapse time of maintenance check j on aircraft k

L_t : Number of maintenance slots per day

p_j : Number of person-hours per day for maintenance check j

P_t : Person-hours 'budget' for day t

Decision Variables:

$m_{k,j,t}$: 1 if aircraft k is doing maintenance check j at time t ; 0 otherwise

$s_{k,j}$: Starting time of maintenance check j for aircraft k

Maintenance Scheduling Model (3)

Objective Function:

$$\min \sum_{k \in K} \sum_{j \in J} [D_{k,j} - s_{k,j}] \times \alpha_j \quad (\text{OF})$$

Subject to:

$$s_{k,j} \leq D_{k,j}, \quad \forall k \in K, j \in J \quad (\text{C1})$$

$$m_{k,j,t} \geq 1, \quad \forall k \in K, j \in J, t \in [s_{k,j}, s_{k,j} + ET_{k,j}] \quad (\text{C2})$$

$$\sum_{k \in K} \sum_{j \in J} m_{k,j,t} \leq L_t, \quad \forall t \in T \quad (\text{C3})$$

$$\sum_{k \in K} \sum_{j \in J} m_{k,j,t} \times p_j \leq P_t, \quad \forall t \in T \quad (\text{C4})$$

$$s_{k,j} \in \mathbb{N}, \quad \forall k \in K, j \in J$$

$$m_{k,j,t} \in [0, 1], \quad \forall k \in K, j \in J, t \in T$$

Other Requirements

However, this model cannot be used in the long-term. Why?

In practice, there are usually other operational requirements that cannot be accommodated in this simple model. Examples:

- No C-checks are allowed during peak-periods (i.e., summer and holiday seasons)
- C-checks stop during weekends and bank holidays
- C-checks cannot start at the same day
- The intervals can be extended (tolerance) but the amount of FH/FC extended needs to be subtracted from following interval

How to address these requirements in an optimisation framework?

Outline - Lecture 9

① Maintenance

- ② a Hangar Checks
- ② b Line Maintenance
- ② c Maintenance Costs

② Maintenance Scheduling Problem

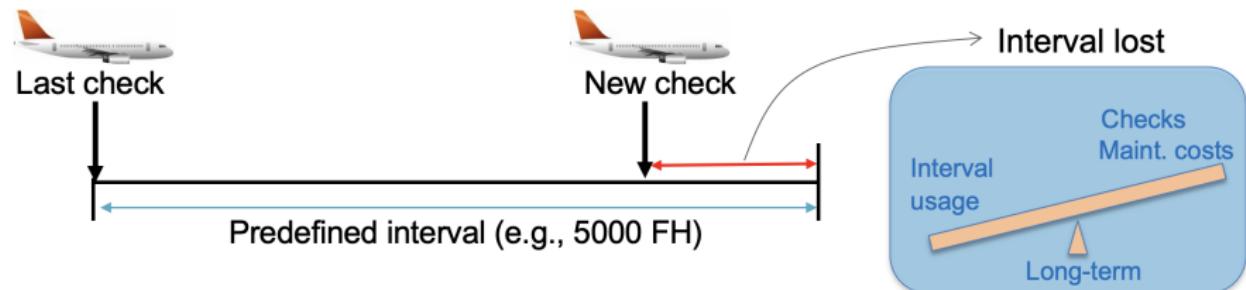
- ③ a Maintenance Scheduling Model
- ③ b Other Requirements

③ Dynamic Programming

- ④ a Challenges
- ④ b A Real-Life Example

Dynamic Programming

The problem: To define the starting (and end) date of a set of maintenance checks that minimize the check interval lost



- To define the long-term hangar checks schedule (3 to 5 years) for a fleet of aircraft

Subject to:

- The given a set of aircraft in a fleet
- The amount of resources available for maintenance (e.g., hangar slots, maintenance teams) every day
- The list of predefined maintenance checks to scheduled, associated with a given duration and **predefined interval**

Dynamic Programming - State

State
Attributes of AC 1
Attributes of AC 2
...
Attributes of AC i
...

Attributes of AC i
A-Check Usage Parameters
C-Check Usage Parameters
D-Check Usage Parameters

AC: Aircraft

DY: Calendar Day

FH: Flight Hours

FC: Flight Cycles

A-Check Usage Parameters

- DY since last A-check
- FH since last A-check
- FC since last A-check

C-Check Usage Parameters

- DY since last C-check
- FH since last C-check
- FC since last C-check

D-Check Usage Parameters

- DY since last D-check

Dynamic Programming - State

State
Attributes of AC 1
Attributes of AC 2
...
Attributes of AC i
...

Attributes of AC i
A-Check Usage Parameters
C-Check Usage Parameters
D-Check Usage Parameters

AC: Aircraft

DY: Calendar Day

FH: Flight Hours

FC: Flight Cycles

A-Check Usage Parameters

- DY since last A-check
- FH since last A-check
- FC since last A-check

C-Check Usage Parameters

- DY since last C-check
- FH since last C-check
- FC since last C-check

D-Check Usage Parameters

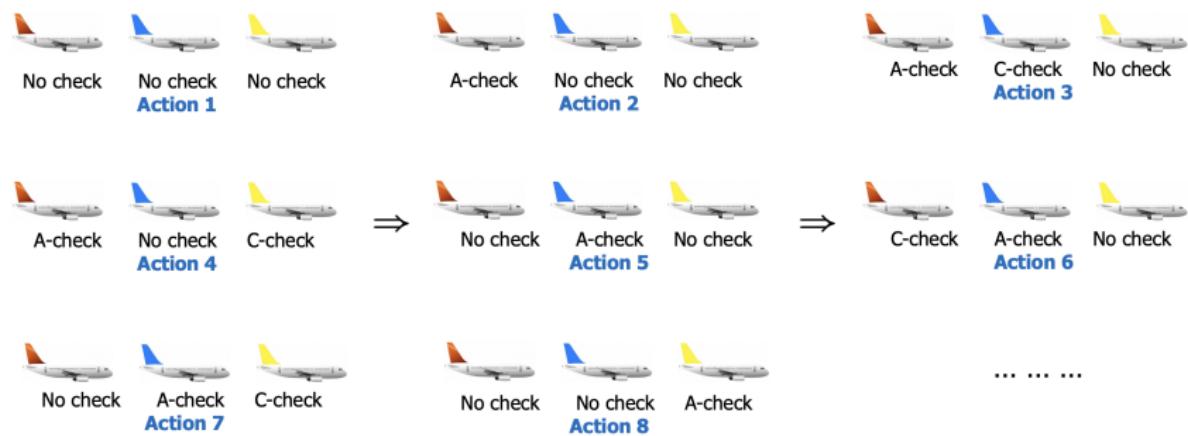
- DY since last D-check

Per aircraft, store 7 attributes. For 50 aircraft, this is 350 features for each state in our dynamic framework!

Additionally, we would like to model day by day!

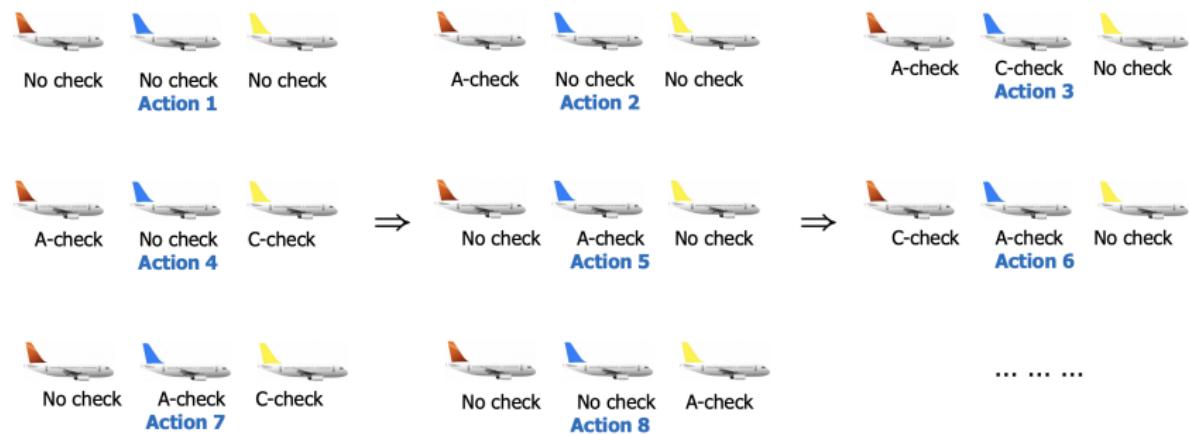
Dynamic Programming - Actions

Example: 3 aircraft, 1 A-check slot, 1 C-check slot



Dynamic Programming - Actions

Example: 3 aircraft, 1 A-check slot, 1 C-check slot



Many actions are possible! All combinations of no check, A-check, and C-check.

Dynamic Programming - Transition Function

1 Aircraft, 1 A-Check slot, 1 C-Check slot, 10 FH/DY, 5 FC/DY

Day t		DY	FH	FC
A-check u. par.		2	20	10
C-check u. par.		400	4000	2000

Day $t + 1$		DY	FH	FC
A-check u. par.		0	0	0
C-check u. par.		401	4000	2000

Day $t + 1$		DY	FH	FC
A-check u. par.		3	20	10
C-check u. par.		0	0	0

Day $t + 1$		DY	FH	FC
A-check u. par.		3	30	15
C-check u. par.		401	4010	2005

(u. par. — usage parameters)

A-Check
C-Check

No Check

Transition function - update of the state (attributes per aircraft) per action performed.

Dynamic Programming – Challenges

What are the main challenges of applying dynamic programming to solve this aircraft maintenance check scheduling problem?

The three curses of dimensionality!

Solution steps:

- ① Multi-dimensional action vector
 - Defining A- and C-check priority - based on deadline of checks
- ② Multi-dimensional outcome sequence
 - Forward induction - do not consider all outcome states, and we do not know the final state
- ③ Multi-dimensional state vector
 - Discretization and state aggregation - visit fewer states
 - Workability check - check feasibility of future states

Step 1 - Maintenance Check Priority



Next C-check in the upcoming 14 days



Next C-check in the upcoming 90 days



Next C-check in the upcoming 5 days

C-Check Priority:

1st



>

2nd



>

3rd



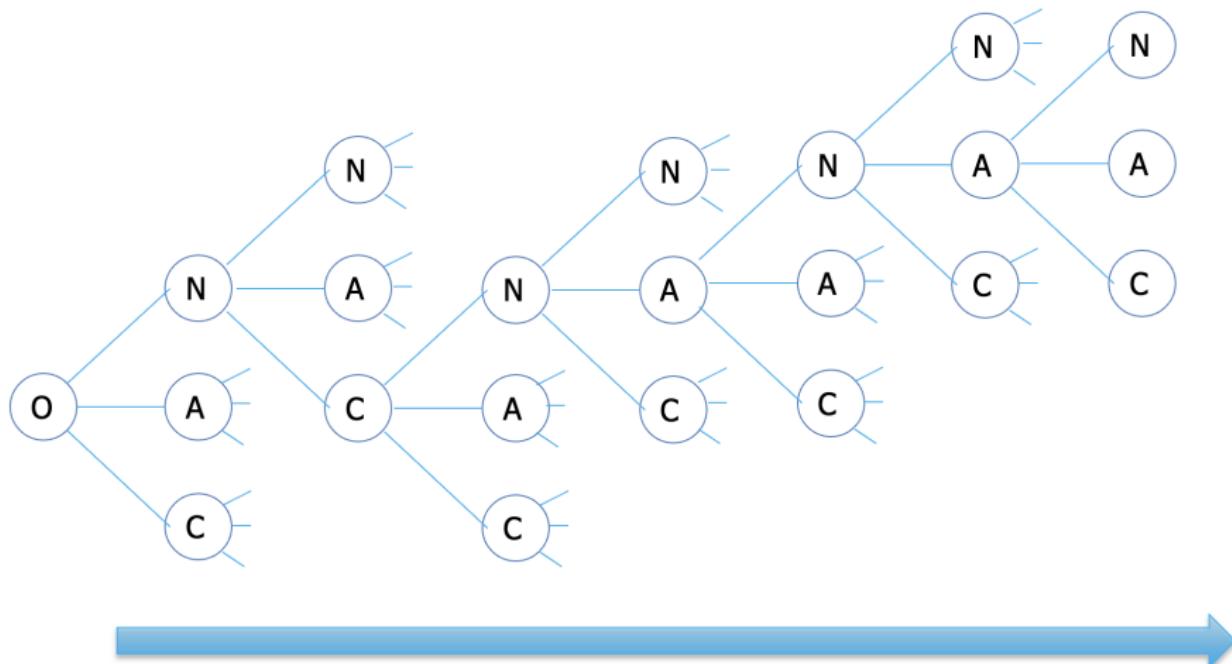
C-check will be scheduled for the **Yellow** aircraft before the C-check for the **Orange** and **Blue** aircraft

Step 2 - Forward Induction

Example: 1 aircraft, 1 A-check slot, 1 C-check slot

N: no check **A**: A-check **C**: C-check

Considering a greedy approach



Step 3 - Discretization & State Aggregation

Attributes of AC <i>i</i>
A-Check Usage Parameters
C-Check Usage Parameters
D-Check Usage Parameters

A-Check Usage Parameters

A-check Interval Utilization

C-Check Usage Parameters

C-check Interval Utilization

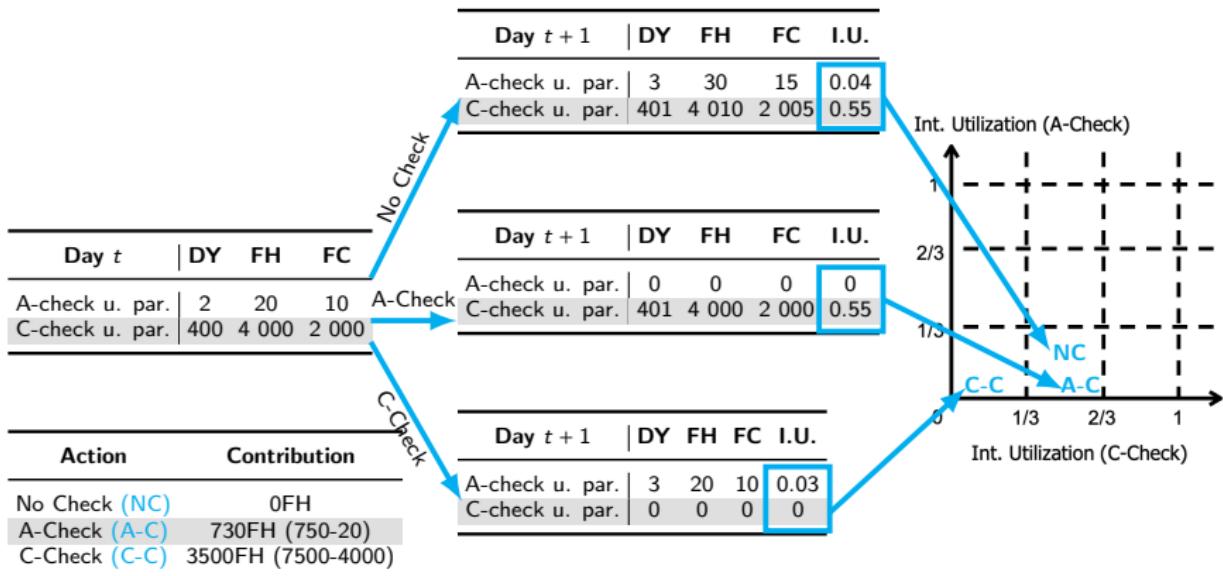
1 Aircraft	DY	FH	FC
Usage Parameters	308	4125	2310
Inspection Interval	730	7500	5000

Aircraft C-check Interval Utilization:

$$\max\left(\frac{308}{730}, \frac{4125}{7500}, \frac{2310}{5000}\right) = 0.55$$

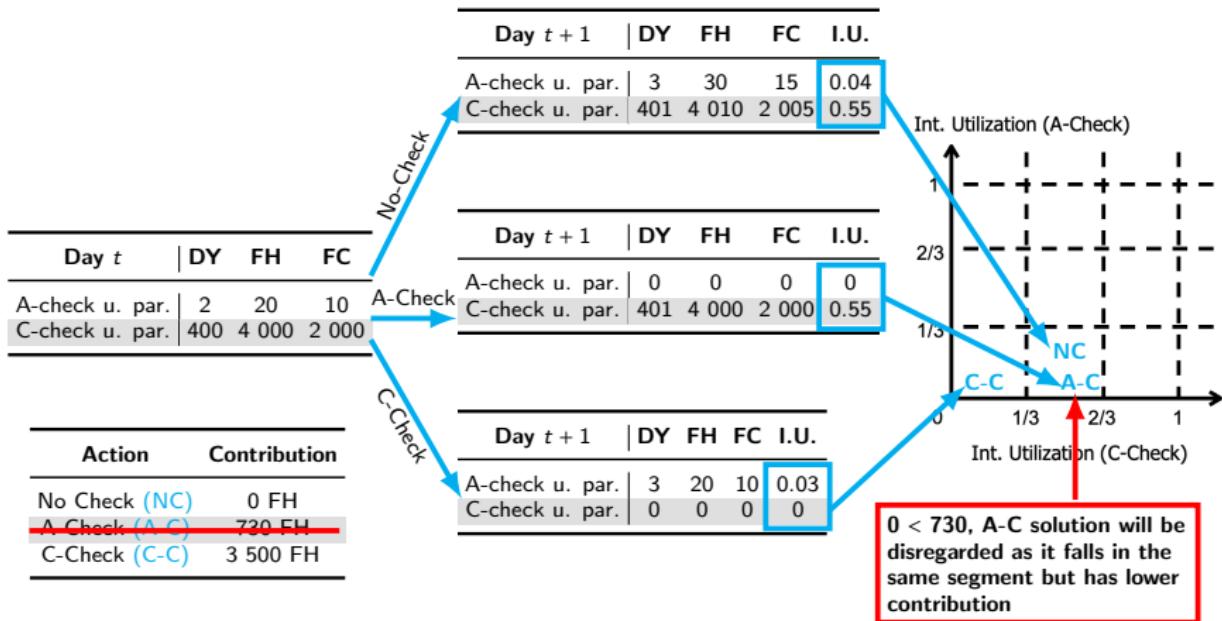
Step 3 - Discretization & State Aggregation (2)

1 Aircraft, 1 A-check slot, 1 C-check slot, 10 FH/DY, 5 FC/DY



Step 3 - Discretization & State Aggregation (3)

1 Aircraft, 1 A-check slot, 1 C-check slot, 10 FH/DY, 5 FC/DY



Step 3 - Workability Check

Issues:

- ① A disregard solution might end up having been the better option - we are compromising the optimality of the solution
- ② Choosing the no-check delays checks; and might result in a unfeasible option in future stages

Solution - to check if a state is feasible in the future:

- Schedule A-/C- check if there is A-/C-check slot
- If no aircraft will be grounded in future stage and waiting for an available A-/C-check slot, we call a state workable
- Only the set of workable states will be used to continue forward induction

A Real-Life Example

- 45 aircraft, AIRBUS A320 Family
- Daily utilization: 10 FH and 5 FC per day
- C-check Interval: 7500 FH / 5000 FC / 730 DY
- C-check can last 7-30 days and A-check lasts 1 day
- A-check interval: 750 FH / 750 FC / 120 DY
- Planning horizon: 5 years
 - 2 A-check slots on Tuesday and Wednesday
 - No A-check slot on weekend and public holidays
 - 3 C-check slots from Oct 1st to May 30th

2018-2021		Airline	DP Based Method	Gain	Estimated Maintenance Costs Savings
C-Check	Avg. FH	≤ 6600	6615.2	$\approx 0.2\%$	300k USD
	Total	≥ 90	88	≈ 2 checks	
A-Check	Avg. FH	≥ 690	717.6	≈ 4 checks	340k USD
	Total	≥ 910	877	≈ 33 checks	
Computation Time		≥ 1 Week	15 min		

Q. Deng, B. F. Santos and R. Curran. A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. European Journal of Operational Research, 2019.

Future Airline Operations with (Hybrid-)Electric and Hydrogen Aircraft

AE4423-Airline Planning & Optimization

Lecture 14

Pieter-Jan Proesmans
January 2024



Do you know this aircraft?

How long can it fly?
What is the charging time?



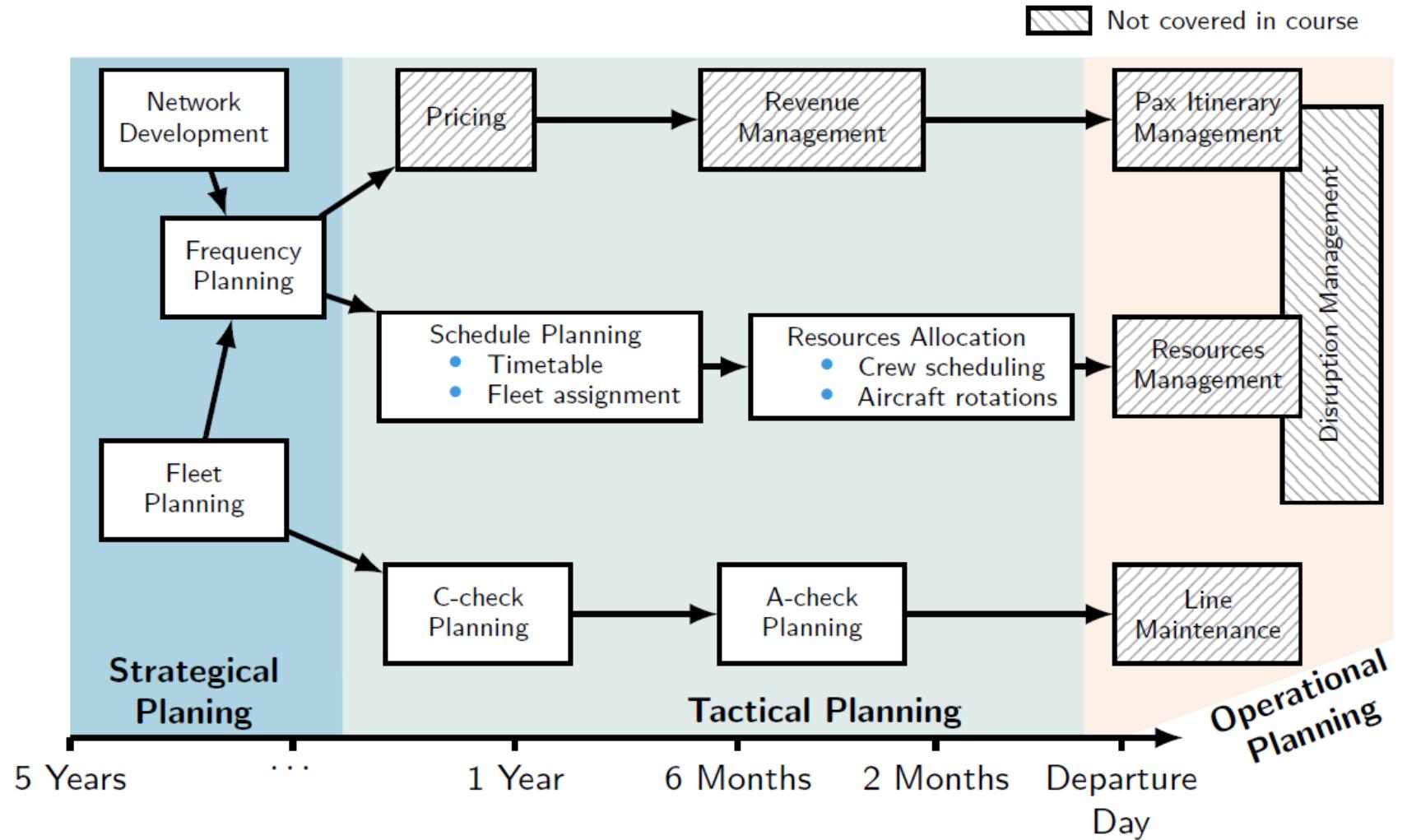
Learning Objectives of This Lecture

After this lecture, you should be able to

1. Explain the challenges of operating aircraft with novel energy carriers
2. Evaluate how these challenges influence airline network development (route & fleet planning)
3. Describe how to extend a strategic fleet planning problem formulation considering these challenges

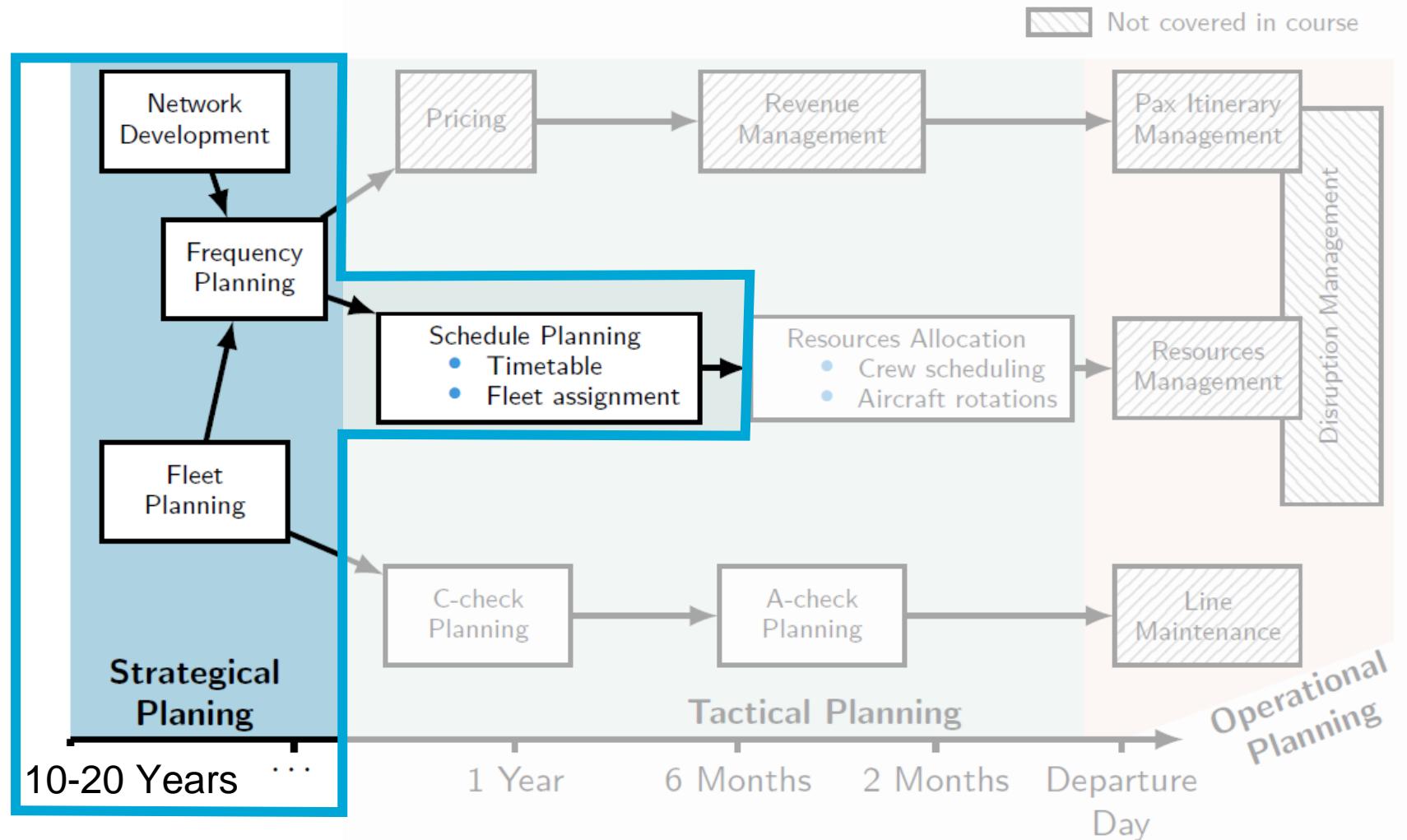
Scope of This Lecture

From Lecture 1:



Scope of This Lecture

This Lecture:



This Lecture

1. Background & Challenges
 - Small exercise for electric aircraft
2. Fleet Planning Problem Formulation
3. Current Research
4. Wrap-Up

Feedback

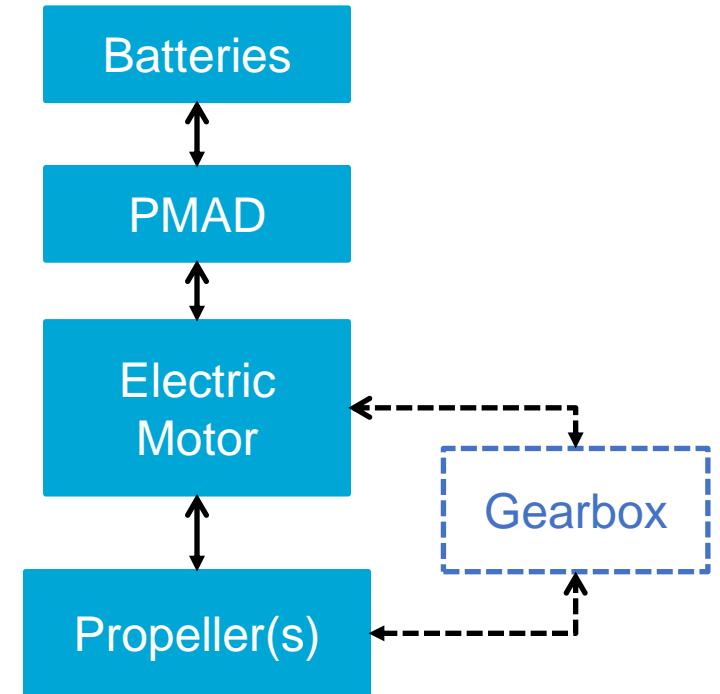
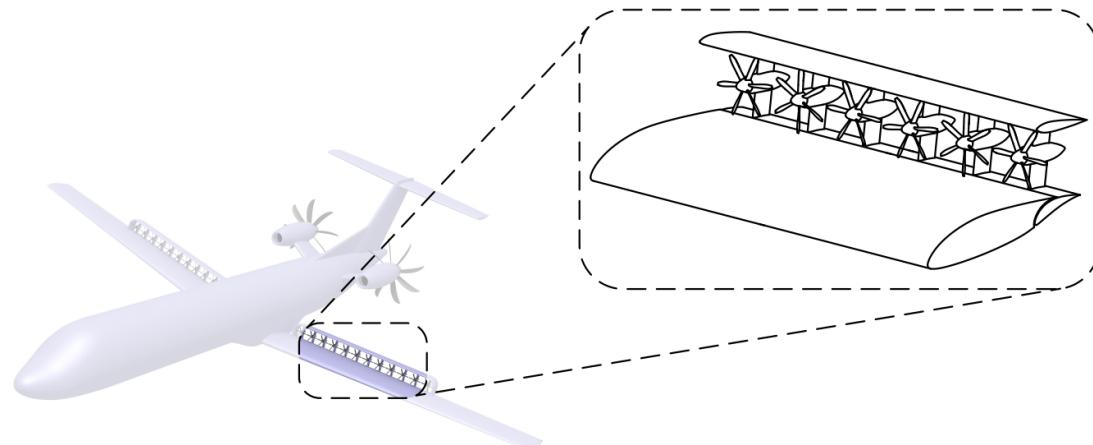
- This session will be recorded
- Please complete the feedback form anonymously
 - Drop
 - Add
 - Keep
 - Improve
- I will collect them in the break

Thank you! ☺

(Hybrid-)Electric and Hydrogen Aircraft: Background & Challenges

Electric Aircraft – Advantages

- Purely driven by batteries
- No in-flight* emissions or contrail formation
- No in-flight* pollution
- New configurations and performance benefits



* Emissions may occur on ground depending on energy source

Electric Aircraft – Charging or Swapping

- Batteries must be recharged or swapped during some turn-arounds

Vehicle or Aircraft	Time to refuel or recharge
Gasoline car	
Electric car	

- Up to 2 times longer turnaround
- Not every airport can recharge

Electric Aircraft – Cruise Speed



Turbofan engines with kerosene combustion

833 km/hr
(Mach 0.78)

20%
slower

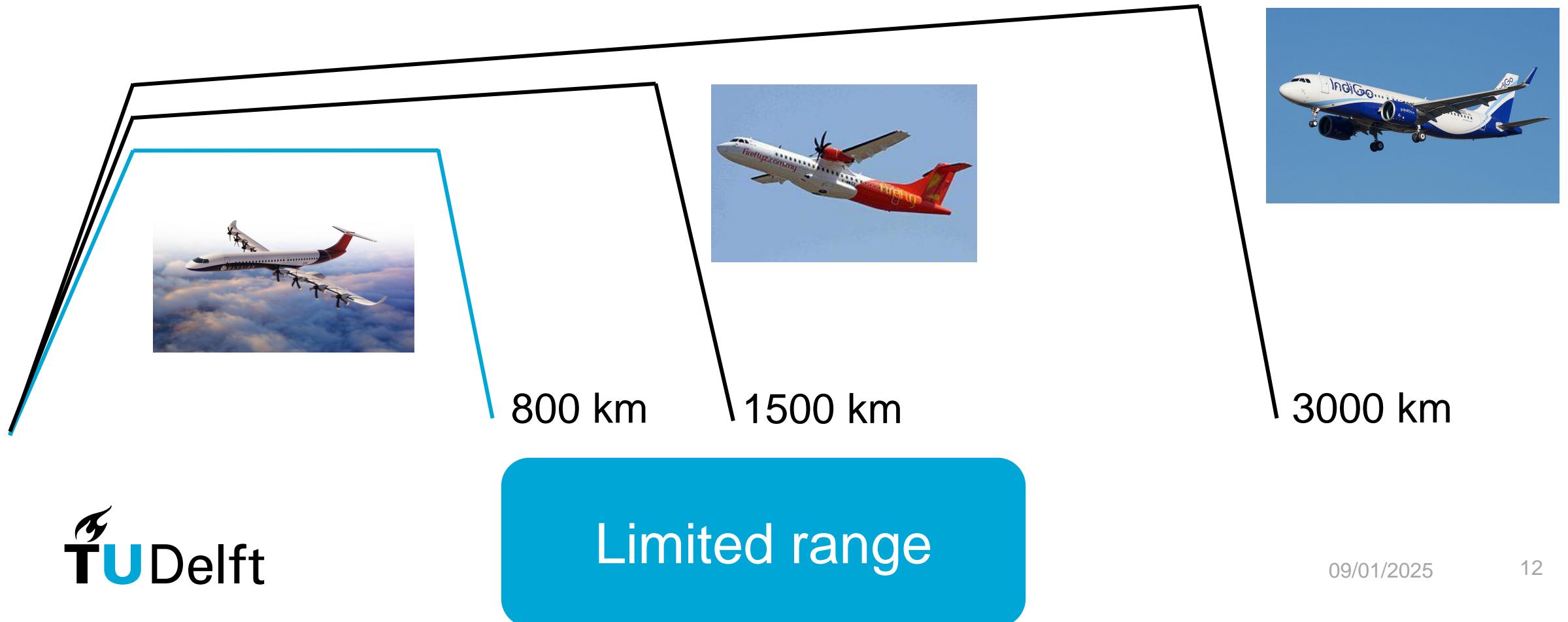
665 km/hr
(Mach 0.60)



Electromotors driving
propellers

Electric Aircraft – Range

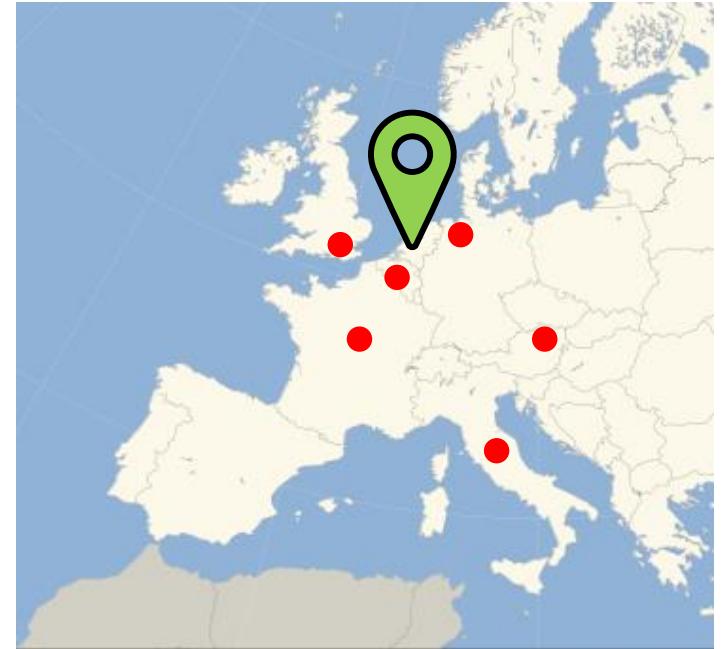
- Batteries are heavier than fuel (energy/kg)



Intermezzo – Find the possible routes



Which of the routes on the sheet can we fly with the electric aircraft?



- Maximum range = 800 km
- Recharge time = 1 hr for 800 km
- Back at hub in less than 3 hours with a flight speed of 600 km/hr

Intermezzo – Find the possible routes

For each route, check if

- 1) the distance can be achieved
- 2) the aircraft can be recharged
- 3) the total operating time is not longer than 3 hours

Done!

You can complete these columns

From	To	Distance	Flight Time	Recharge time	Charging available?	Range ok?	Total operating time	Time ok?	Flight ok?
Hub	Dest.	D	FT	RT	Yes or No	$D < 800?$	$TOT = RT + 2 \times FT$	$TOT < 3?$	Yes or no?

- 4) Say if the route can be flown based on range, time, and charging infrastructure



0 response submitted

Which routes can this electric aircraft fly?

Scan the QR or use
link to join



<https://forms.office.com/e/PGnDjiVhMT>

Copy link

Brussel

Hamburg

Paris

Rome

London

Vienne



1 of 1



Show correct answer

Intermezzo – Find the possible routes

From	To	Charging available?	Charging time	Range ok?	Total operating time	Time ok?	Flight ok?	Reason
Ams	Hamburg	Yes	0.46	Yes!	1.7	Yes	Yes ☺	
Ams	Paris	Yes	0.54	Yes!	2.0	Yes	Yes ☺	
Ams	London	Yes	0.45	Yes!	1.7	Yes	Yes ☺	



Can we make the Brussels route work?

Hybrid-Electric Aircraft

- Provide more range and flexibility than pure electric aircraft
- Different powertrains exist
- Supplied power ratio

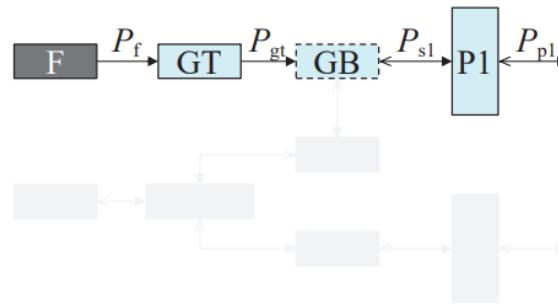
$$\phi = \frac{P_{bat}}{P_{bat} + P_{fuel}}$$

where

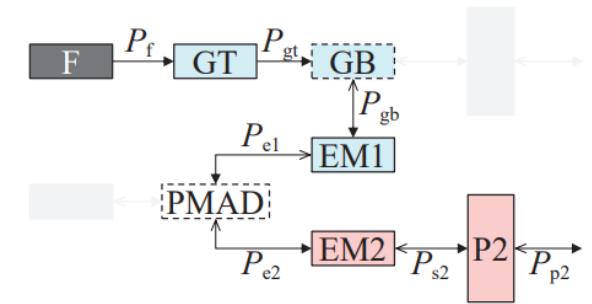
P_{bat} = Battery power

P_{fuel} = Fuel power

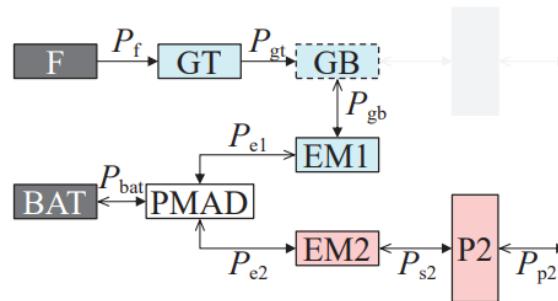
1. Conventional



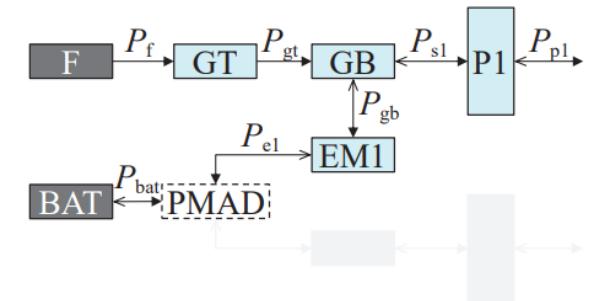
2. Turboelectric



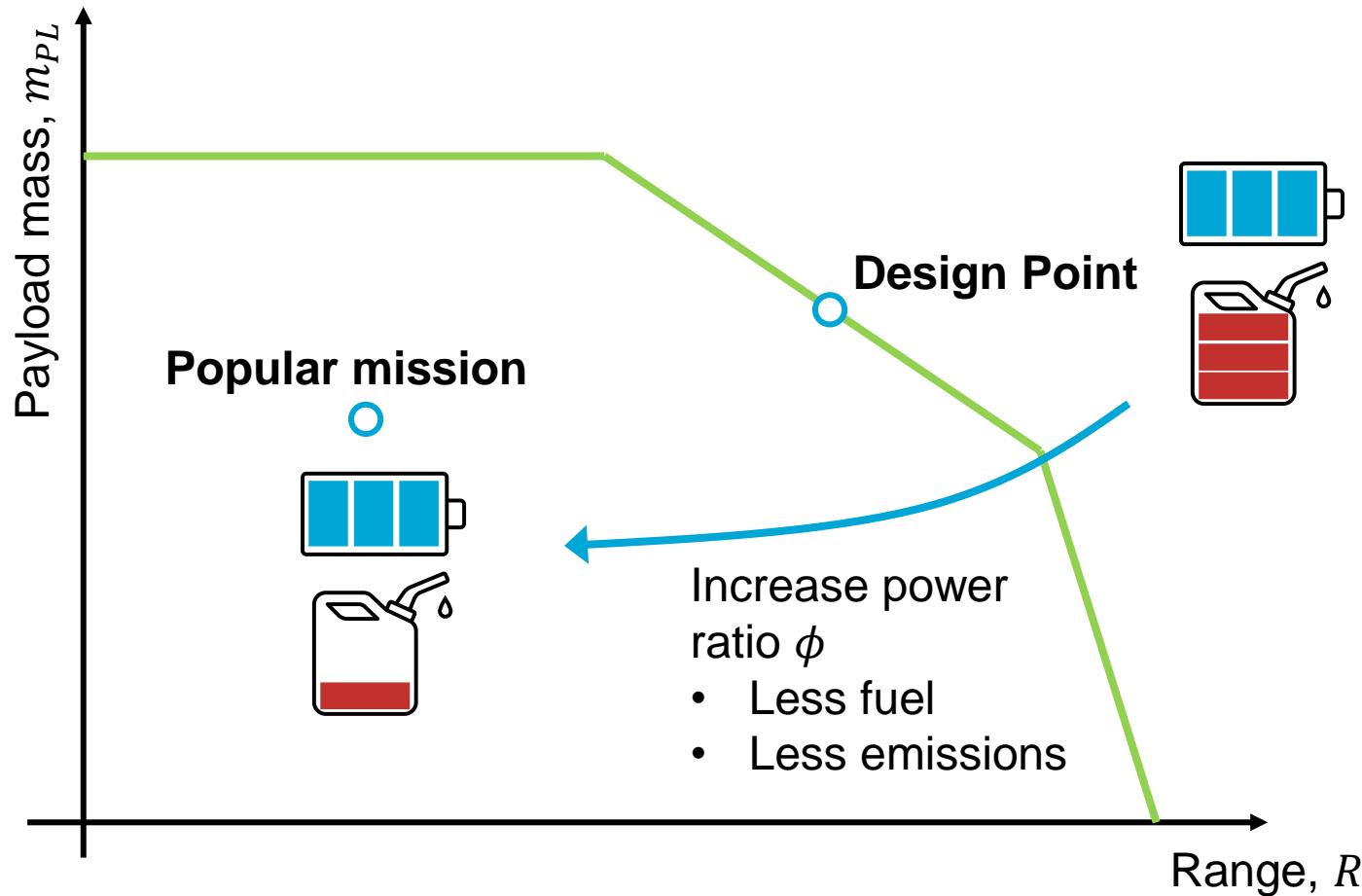
3. Serial



4. Parallel



Hybrid-Electric Aircraft

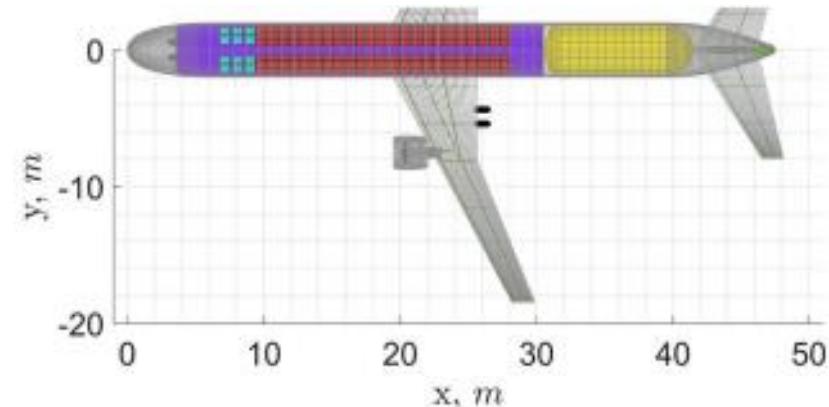


- More flexibility in
 - Range
 - Charging vs. refueling
- ϕ is both a design and control variable

Hydrogen Aircraft

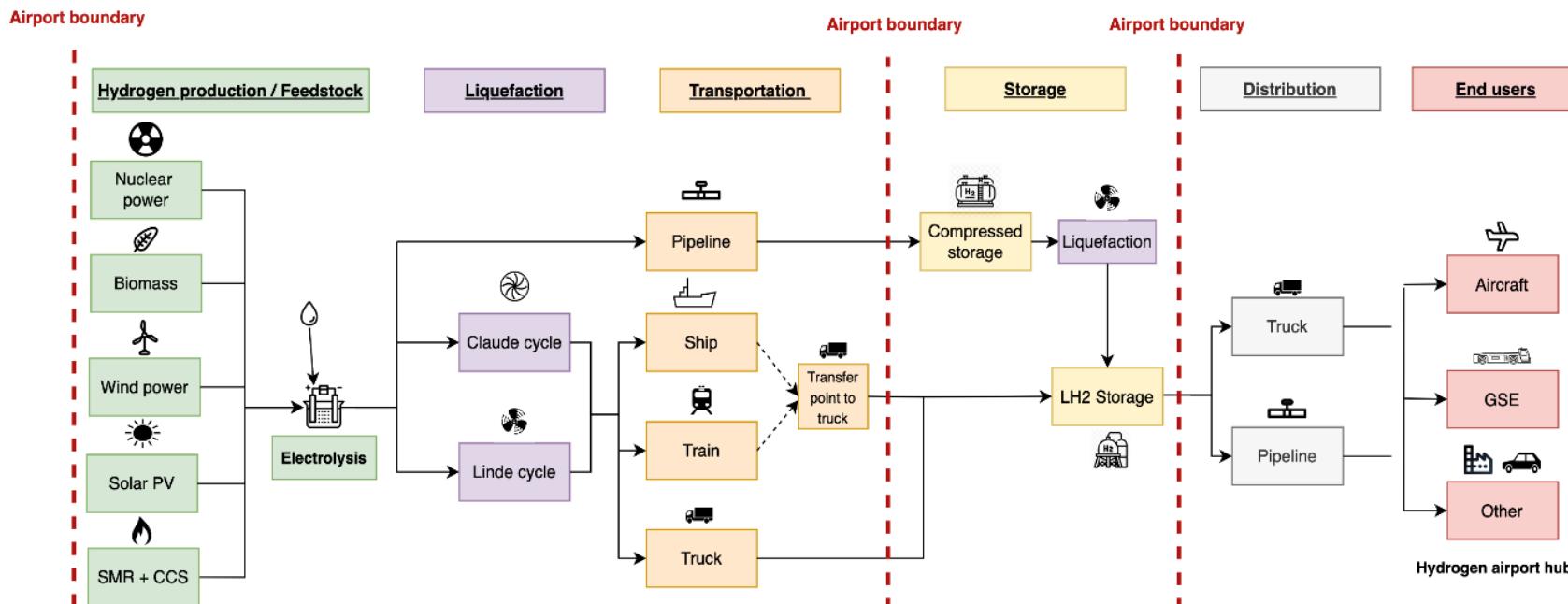
- Provide more range than electric aircraft
- CO₂ emissions are eliminated, some non-CO₂ remain
- Cryogenic tank integration can lead to higher energy consumption than kerosene alternative, depending on the size and integration

Power train	Emission Species
H ₂ Fuel Cells	- Water vapor - Contrails
H ₂ Combustion	- Water vapor - Contrails - Nitrogen oxides (NO _x)



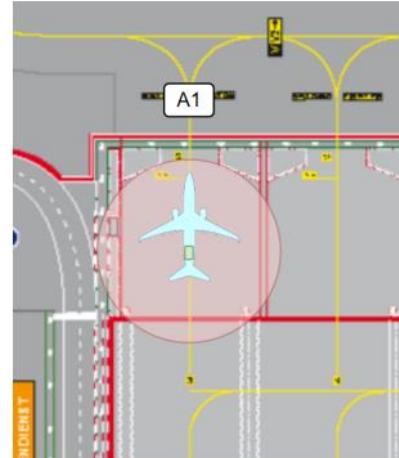
Hydrogen Aircraft – Infrastructure

- Refuelling liquid hydrogen requires new, expensive infrastructure
- Supply to airports can be complex depending on airport location
- Hydrogen for aviation will initially be more expensive than kerosene

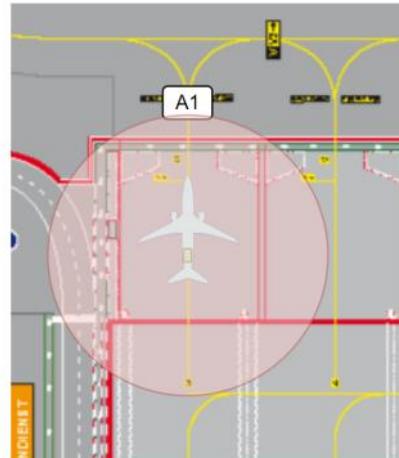


Hydrogen Aircraft – Refuelling

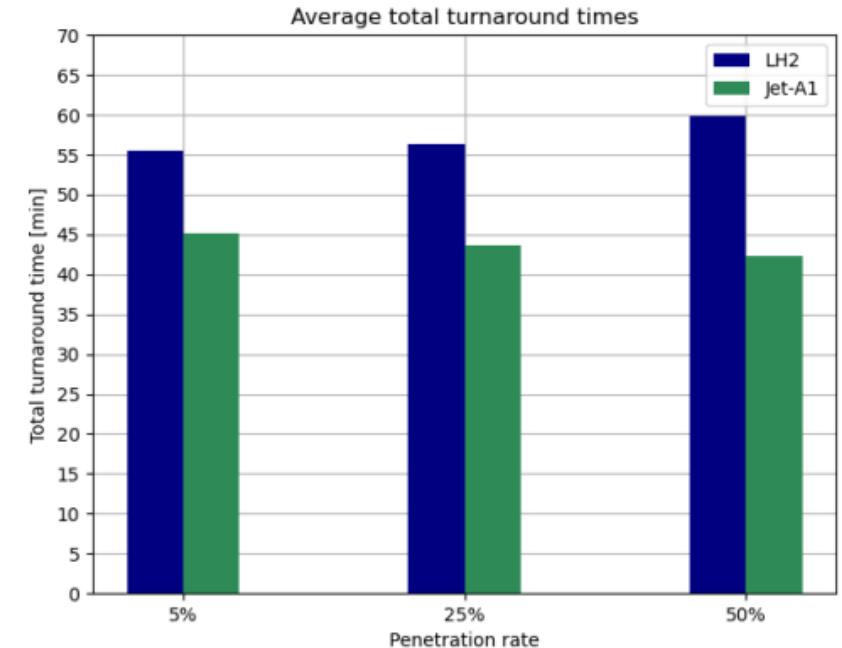
- Ground operations for turnaround may be slower
- For liquid hydrogen
 - Fuel lines need to be purged and cooled
 - Flow rates are still uncertain
- Safety measures around refueling process



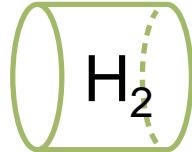
((b)) Safety zone radius of 30 meter.



((c)) Safety zone radius of 45 meter.



Summary of Key Operational Challenges

- Electric 
 - Slower cruise flight than turbofan aircraft
 - Only regional range
 - Recharging or battery swapping infrastructure & time
 - Hydrogen 
 - Regional to medium range, but not long range
 - High energy cost
 - Refueling infrastructure & time
-]
- Hybrid powertrains provide a trade-off
- More flexibility
 - Non-zero emissions

Formulate the Fleeting Planning Problem

Modelling – Fleet & Network

- Objective function to maximize profit = revenue - cost

$$\text{Revenue} = \sum_{r \in R} \sum_{i \in N} \sum_{j \in N} \left(yield_{ij} \cdot d_{ij} \cdot x_{ij}^r + \sum_{m \in R} yield_{ij} \cdot d_{ij} \cdot w_{ab}^{rm} \right)$$

$$\text{Cost} = \sum_{k \in K} \left[\sum_{r \in R} (cost_k^r \cdot z_k^r) + ownership_k \cdot ac_k \right]$$

- Cost per route now also includes a carbon tax

$$cost_k^r = trip_k + hourly_k^r + fuel_k^r + electricity_k^r + \boxed{tax \cdot emission_k^r}$$

Modelling – Fleet & Network

- New set R , indexed by

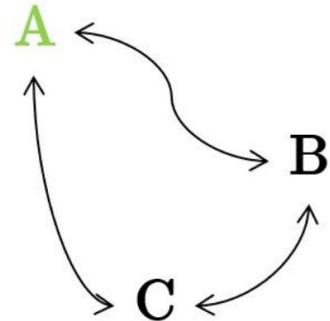
- r for a route
- m for a transfer route

- Newly defined decision variables

- x_{ij}^r : Number of passengers travelling from i to j on route r
- w_{ij}^{rm} : Number of passengers travelling from i to j on route r followed by route m
- z_{ij}^r : Frequency of aircraft k on route m

Single flights

$A \leftrightharpoons B$
 $B \leftrightharpoons C$
 $A \leftrightharpoons C$



Routes

$A \leftrightharpoons B \leftrightharpoons A$
 $A \leftrightharpoons C \leftrightharpoons A$
 $A \leftrightharpoons B \leftrightharpoons C \leftrightharpoons A$
 $A \leftrightharpoons B \leftrightharpoons C \leftrightharpoons B \leftrightharpoons A$
 $A \leftrightharpoons C \leftrightharpoons B \leftrightharpoons C \leftrightharpoons A$
etc.

Modelling – Fleet & Network

- Turnaround time: modelled in a similar manner as in the assignment
- Range

$$z_k^r \leq a_k^r \quad \forall k \in K, r \in R \longrightarrow a_k^r = \begin{cases} 10000 & \text{if total route distance is smaller than range} \\ 0 & \text{otherwise} \end{cases}$$

- Charging or refuelling at limited number of airports

$$\sum_{r \in R} \left(\sum_{i \in N} R1_{ic}^r \cdot z_k^r \right) = \sum_{m \in R} \left(\sum_{j \in N} R1_{cj}^r \cdot z_k^m \right) \quad \forall k \in K, c \in N_{charging}$$

where $R1$ is a binary parameter which is 1 if flight from i to j is part of route r .

Start and end of routes should occur at an airport with charging capabilities.

Current Research

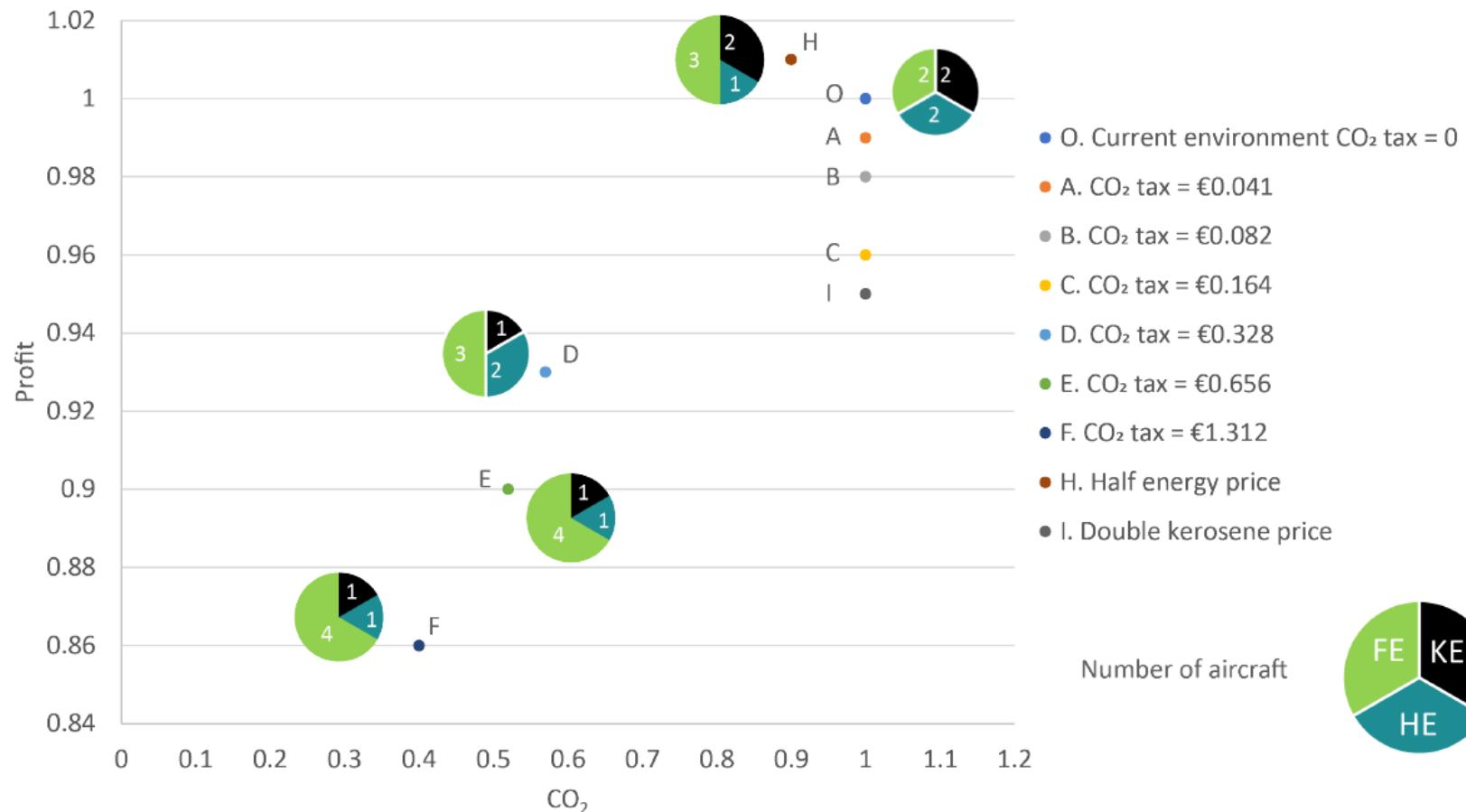
1. Coupled Strategic Planning & Design
2. Reflexive Demand
3. Hydrogen Tankering

Current Research Questions

- Under what technical and economical assumptions will novel aircraft be assigned in a network?
- What strategies can be employed to help the entry into service?
- How will the demand respond to these new technologies?
- What are the optimal aircraft designs in terms of range and payload?
- ...

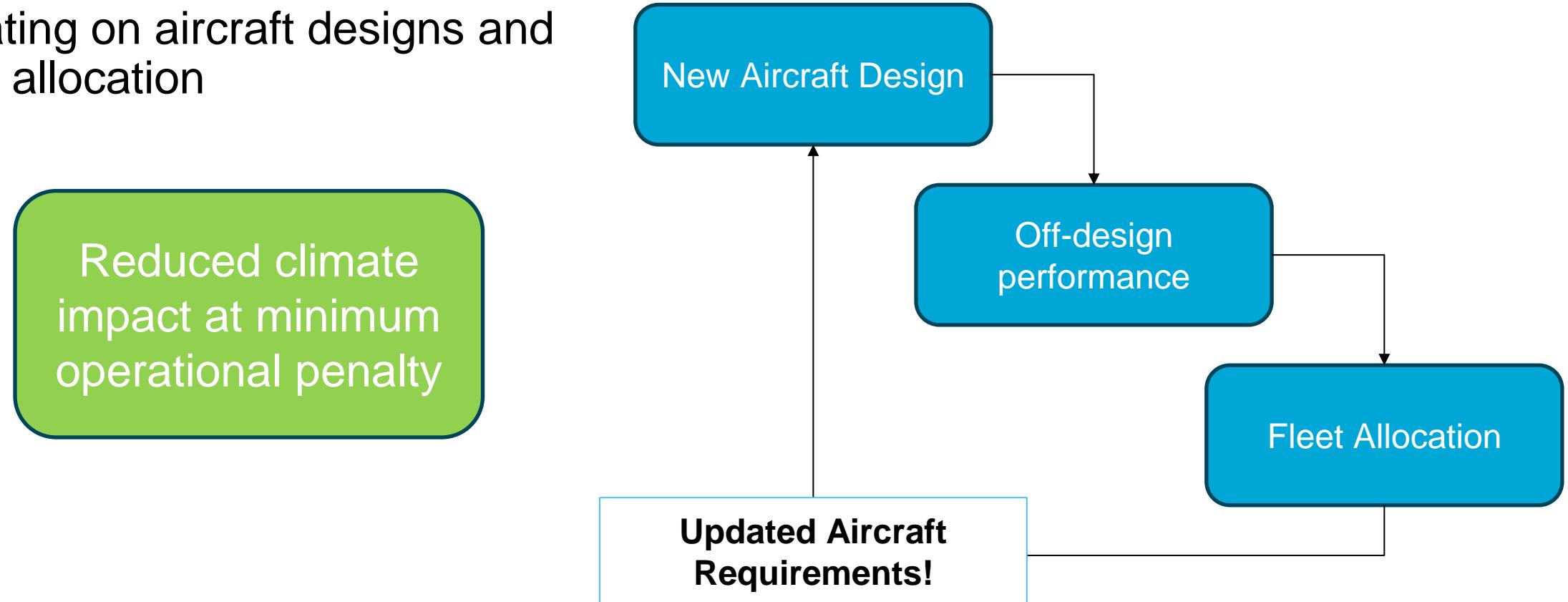
Coupled Strategic Planning & Design

- Impact of CO₂ taxation on strategic fleet planning
- Case study for the Azores



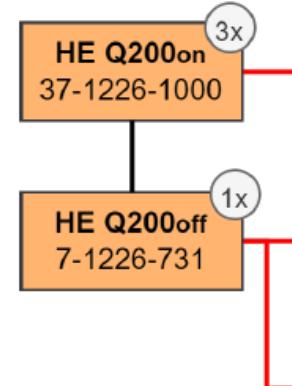
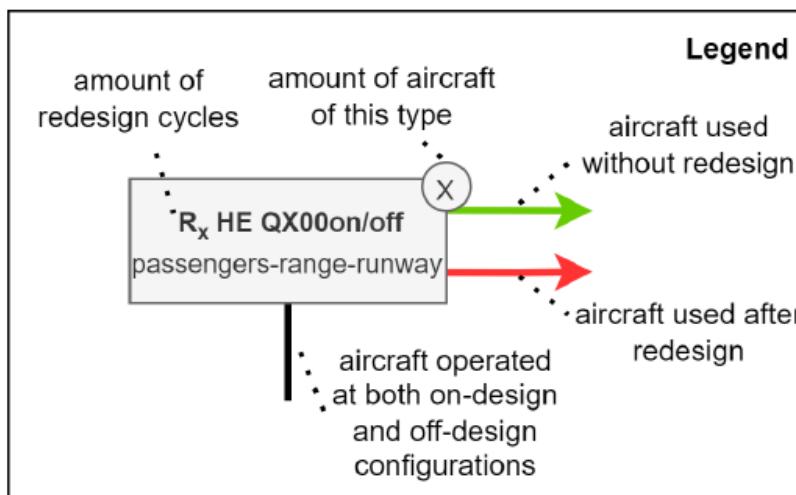
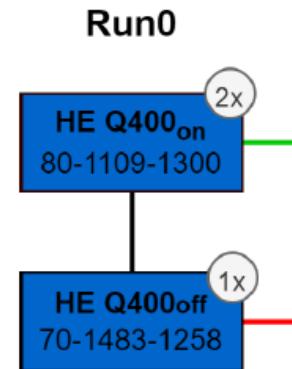
Coupled Strategic Planning & Design

Iterating on aircraft designs and fleet allocation



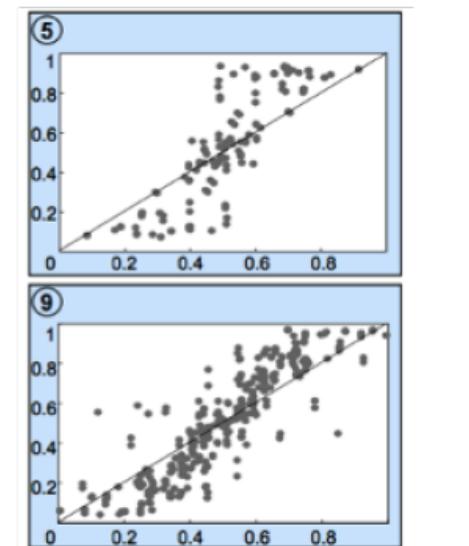
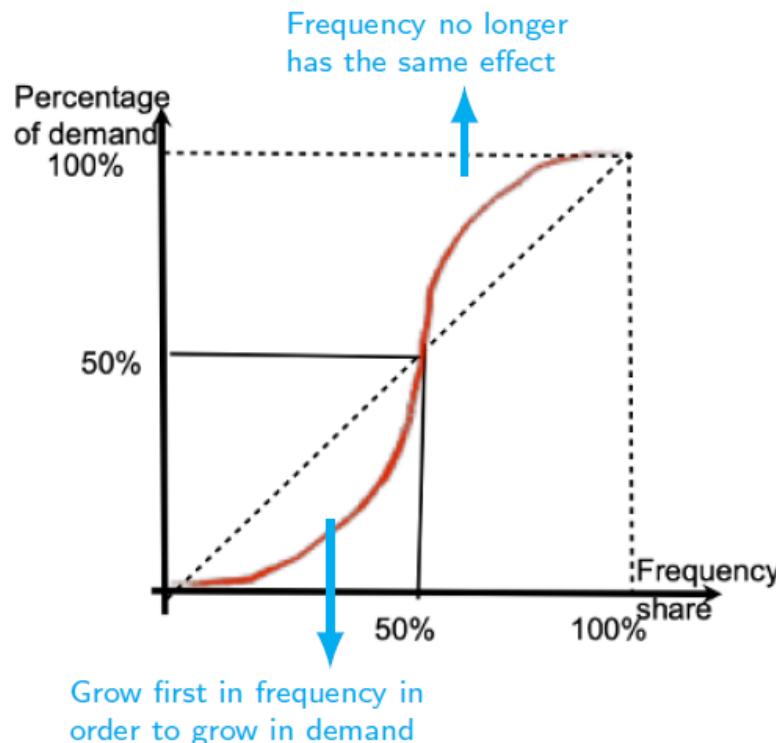
Coupled Strategic Planning & Design

Iterating on aircraft designs and fleet allocation



Strategic Planning with Reflexive Demand

- Remember from Lecture 1 – Market share depends on frequency, price, and quality of service



New aircraft with lower flight frequency (due to increased TAT or reduced cruise velocity) will capture less demand.

Strategic Planning with Reflexive Demand

Model strategic planning decisions with:

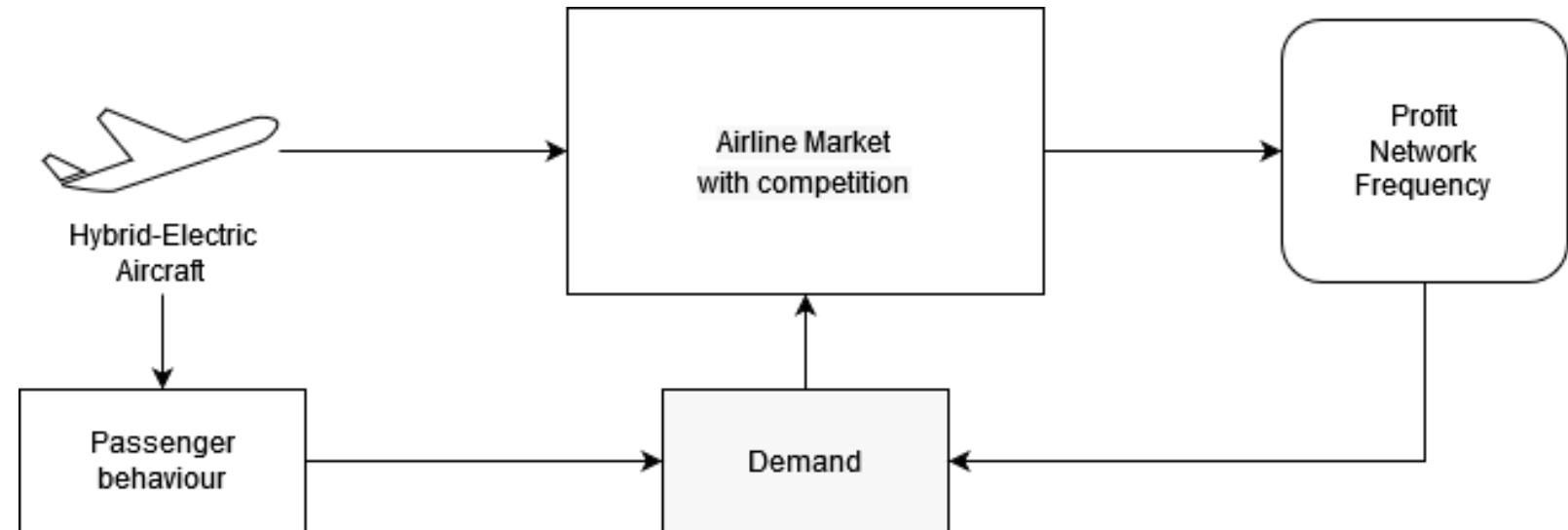
- Supply-dependent demand
- Competing airlines

Considering:

- Different aircraft properties
- Cost and revenue changes
- Airport compatibility

To study the effect on:

- Fleet composition
- Aircraft utilization
- Overall demand
- Market share



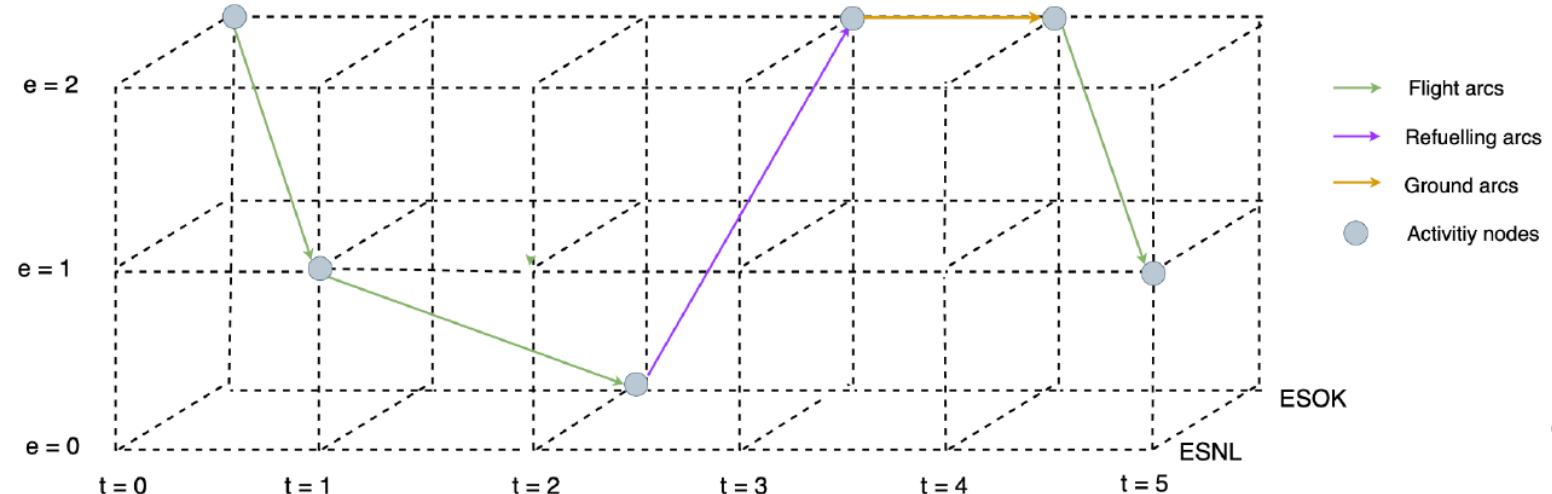
Hydrogen Aircraft Tankering

- Not every airport will be ready at the start
 - Which airports should adapt first?
 - How to operate the aircraft?

Carry more fuel on board to eliminate refuelling at destination(s) and performing multiple flights with 1 full tank

**Not sustainable
for kerosene
aircraft! Why?**

- Considering a time-space-energy network



Ongoing MSc Thesis Projects

- Hybrid-Electric aircraft allocation with partial recharging
- Network development for electric aircraft
- Fleet allocation of climate-optimal aircraft
- Fleet development model for hydrogen aircraft

Let us know if you
are interested!

Wrap-Up

Summary of This Lecture

- Influence of novel energy carriers on airline operations
 - Reduced range capability
 - Higher fuel or energy cost (in particular for hydrogen)
 - Revised turn-around processes, with possibly longer TAT
- Reformulate fleet planning model to include these restrictions
- Examples of ongoing research and interactions with other disciplines

Learning Objectives of This Lecture

After this lecture, you should be able to

1. Explain the challenges of operating aircraft with novel energy carriers
2. Evaluate how these challenges influence airline network development (route & fleet planning)
3. Describe how to extend a strategic fleet planning problem formulation considering these challenges

Questions Outside the Scope

- Maintenance of novel energy carriers
 - Health monitoring (e.g., battery state of health)
 - Maintenance scheduling
- Crew scheduling
- Supply of sustainable energy
- Policies and investments
- ...

Thank you for your
attention and feedback!
Any Questions?

Back Up Slides

Hybrid-Electric Off-Design Performance

