

Dominance and Indifference in Airline Crew Scheduling

Amy Mainville Cohn, KoMing Liu, and
Shervin Ahmadbeygi

University of Michigan IOE Department

October 20, 2003

Funded in part by a University of Michigan Rackham Research Grant

Motivation

- Many exciting problems in the airline industry today build on “old-fashioned” core problems
 - ◆ More accurate basic planning models (e.g. Barnhart, Kniker, and Lohatepanont 2002)
 - ◆ Integrated models (e.g. Cordeau et al 2000)
 - ◆ Real-time recovery systems (e.g. Rosenberger, Johnson, and Nemhauser 2001)
 - ◆ Robust approaches (e.g. Yen and Birge 2000)
- These core problems are larger than ever before and must be solved faster than ever before!

Motivation, cont.

- Our original goal was to integrate crew scheduling and fleet assignment
- Crew problem alone becomes much harder because it's all flights, not just one fleet type at a time
- So for the near future, we are focusing just on crew scheduling alone...
 - ◆ ...just on crew pairing alone...
 - ◆ ...just on the pricing problem for crew pairing alone...

The Crew Pairing Problem

- A *pairing* is a sequence of flights that can be flown by a single crew over a series of days, starting and ending at their home base
- Has to satisfy many rules dictated by labor negotiations and FAA regulations
 - ◆ Max flying time
 - ◆ Max time away from base
 - ◆ Min time between flights
 - ◆ Min rest
- Has a non-linear cost which is a function of flying time, time-away-from-base, idle time, etc.

The Crew Pairing Problem, cont.

- Avoid non-linearities with a set partitioning formulation:
 - ◆ x_p is a binary variable that takes value 1 if pairing p is included in the solution, else 0
 - ◆ δ_{fp} is a constant with value 1 if flight f is included in pairing p , else 0
 - ◆ c_p is the cost of pairing p

The Crew Pairing Formulation

$$\text{Min } \sum_p c_p x_p$$

$$\text{St } \sum_p \delta_{fp} x_p = 1 \quad \forall f$$

$$x_p \in \{0, 1\} \quad \forall p$$

Solving the Crew Pairing Problem

- Can't write out the SP model explicitly – hundreds of billions of pairings (or more!)
- Solve using *branch-and-price*
 - ◆ Solve the IP using branch-and-bound
 - ◆ Solve each node of the branch-and-bound tree using *column generation*
 - ◆ Instead of explicitly computing the reduced cost of each non-basic variable, solve an optimization problem – *pricing problem* – to identify the most negative reduced cost column

Generating Pairings

- One of the key challenges is how to find negative reduced cost pairings
 - ◆ All of the non-linearities that we eliminated from the master problem are now in the pricing problem!
- Many interesting approaches for trying to find good pairings
 - ◆ Randomized approaches
 - ◆ Parallel algorithms
 - ◆ Approximations

Generating Pairings, cont.

- Predominant approach in the literature to find *provably optimal* solutions (or valid lower bounds) is *multi-label shortest path*
 - ◆ A pairing corresponds to a path in a flight network
 - ◆ The reduced cost of a path is its true cost minus the sum of the duals associated with the cover constraints for the flights in that path

Shortest Path Approaches

- Shortest path approaches are enumeration-based approaches that reduce the search space by recognizing *dominance*
- In a traditional shortest path problem
 - ◆ If two path fragments both lead to the same point, they both have the same opportunities for completing the path
 - ◆ Therefore, we can throw away the longer fragment – it is *dominated*, i.e. will always lead to a sub-optimal solution

Multi-Label Shortest Path

- We can't do this in the crew pairing pricing problem
 - ◆ Two fragments connecting the same start/end points don't necessarily have the same opportunities
 - ◆ Shorter fragment (i.e. one with most negative reduced cost) might use more flying time, eg. – therefore would have fewer remaining options
 - ◆ Only prune if dominance in all categories – eg. lower cost, less flying time, less time away from base, etc.

Challenges of MLSP

- As the number of labels grows, it's harder to establish dominance and thus we enumerate larger and larger percent of the paths
- Furthermore, we claim that the use of dominance in MLSP can in fact lead to sub-optimality
 - ◆ Because flights cannot be repeated in a pairing, no fragment can ever truly dominate another – will always be lacking opportunities based on flights already covered

Example of MLSP Sub-Optimality

Flight	Orig	Dest	Dept	Arr	FlyTime
A	1	2	9	11	2
B	2	3	13	15	2
C	3	2	6	12	6
D	2	4	16	18	2
E	4	1	10	12	2
F	1	2	13	15	2
G	2	3	14	17	3
H	3	1	22	26	4

Example of MLSP Sub-Optimality

Min duty pay 4

f .6

Max fly time 8

Max duty time 12

Max # duties 3

Min turn .5

Crew bases Station 1

Example of MLSP Sub-Optimality

- This problem has four feasible solutions:
 - ◆ Pairings AB|C|GH and FD|E with cost 25.2
 - ◆ Pairings AB|CD|E and F|GH with cost 27.2
 - ◆ Pairings AG|CB|H and FD|E with cost 25
 - ◆ Pairings AG|CD|E and F|B|H with cost 29

Example of MLSP Sub-Optimality

- Pairing AG|CB|H is part of the unique optimal solution
- Suppose that this is the only pairing *not* included in the restricted master – the pricing problem should identify this as the candidate pivot variable
- However, the sequence of flights AB|C dominates the sequence AG|C – it has lower cost, lower reduced cost (based on the current dual values), less flying time, and the same start/end points
- We therefore discard AG|C because it's dominated; but AB|C does not lead to a feasible negative reduced cost pairing – this satisfies the optimality criteria!

Search Methods

- Optimization techniques rely primarily on finding ways to disregard portions of the feasible region
 - ◆ Simplex only looks at “corners”
- In combinatorial (i.e. discrete) problems, we often rely on *dominance* and *indifference*, in conjunction with lower bounds using dual information, to prune sub-optimal regions

Dominance and Indifference

- In airline planning, we often select and assemble building blocks
 - ◆ Certain blocks may dominate others
 - ◆ We may be indifferent to the different blocks within a set
 - ◆ Given the blocks, we may be indifferent to how they are combined, or some combinations may dominate others

Decomposition

- This often allows us to decompose our problem
 - ◆ First, determine whether a given subset of decisions is part of our solution
 - ◆ Second, determine which decisions to make within the chosen subsets
- Potential improvements in tractability
 - ◆ First stage has decreased in scope
 - ◆ Second stage is a feasibility problem rather than an optimization problem

Some Examples

- Rexing et al 2000 integrate schedule design and fleet assignment
 - ◆ Allow flight times to shift
 - ◆ First stage: assign flights to fleet types and to limited windows of time
 - ◆ Second stage: assign specific departure times within these windows
- Cohn and Barnhart 2003 integrate crew pairing and maintenance routing
 - ◆ Exploit the fact that only a small subset of aircraft turns have impact on crew decisions
 - ◆ First stage: choose crew pairings and only relevant aircraft turns
 - ◆ Second stage: construct a maintenance solution containing these turns

Dominance in Crew Pairing

- One way to limit the search space when generating pairings is to eliminate dominated fragments
- Difficulties in a path-generating approach:
 - ◆ If we have limited dominance we enumerate most of the feasible region
 - ◆ If we have lots of dominance we run the risk of missing relevant pairings
- Can we use different forms of dominance/indifference to identify negative reduced cost pairings?

Alternate Strategy

- The cost of a pairing is a function of
 - ◆ Duty costs
 - ◆ Number of duties
 - ◆ TAFB
- The cost of a duty is a function of
 - ◆ Flying time
 - ◆ Elapsed time
 - ◆ Minimum duty pay
- Thus, the cost of a pairing is largely determined by the first and last flight of the duties that comprise it

Alternate Strategy

- A *pairing framework* is a set of flights representing the starting and ending flights of each duty in that pairing
- The number of pairing frameworks is much smaller than the number of pairings
 - ◆ Eg. In one major U.S. carrier's network, two flights can “span” as many as 700 distinct duties!
- Instead of reducing search space by disregarding dominated pairing fragments, throw away unnecessary pairing frameworks; expand remaining frameworks to check for negative pairings

Viable Pairing Frameworks

- A pairing framework is said to be *viable* if we can't find a non-negative lower bound on the reduced cost of the corresponding pairings
- Thus we want to generate tight lower bounds to leave as few frameworks viable as possible
- Lower bound on duty cost stems from shortest-path between the spanning flights
- Upper bound on dual contribution stems from longest-path between the spanning flights
- Both can be determined once a priori – at most $|F|^2$
- Currently testing with real-world data to see if initial bounds are tight enough or if they need to be strengthened further

Conclusions/Future Research

- Currently testing this alternative pricing approach – may require enhancements to tighten the bounding of the dual contribution
- This may lead us to other approaches as well
- Key point: We need to expand beyond path-based enumeration to solve very large crew problems very quickly, and in fast can benefit from improvements to all core problems!