Lecture Notes in Management Science (2016) Vol. 8, 54–58 ISSN 2008-0050 (Print), ISSN 1927-0097 (Online)

Robust itinerary generation for disruption management in airlift operations

Udatta S. Palekar [1] and Alok Tiwari [1]
[1] University of Illinois at Urbana-Champaign, Urbana, USA
{palekar, tiwari2}@illinois.edu

Proc. ICAOR 2016
Rotterdam, The Netherlands

Keywords:
Airlift disruption management
Dynamic replanning
Itinerary generation
Robust optimization

Abstract

We describe an approach for generating robust itineraries for disruption management in airlift operations. The concept of a robust optimal policy and a robust worst-case itinerary are developed. A polynomial algorithm that calculates both the robust policy and worst-case itinerary for a discretized version of the problem is presented along with preliminary computational results for a small problem.

Introduction

In this article we consider an aspect of the airlift mission re-planning problems. In such problems there is already a planned itinerary which faces disruptions. Such problems are faced by courier delivery services, commercial airlines, and the US Air Mobility Command when planning and executing the movement of personnel and equipment for the US Department of Defense. Some of the challenging features of these missions include demand uncertainties and mission disruptions and delays caused by weather events, breakdowns, or changes necessitated for command and control. In various formulations of the airlift problems the key decision variables involve the choice of appropriate 'itineraries' from a pre-generated collection and the objective is to achieve the best performance on average. We define these terms more precisely in a later section.

In the present article we generalize the notion of an itinerary to that of a 'policy' and describe a method for generating policies that are robust in the face of disruptive events and deliver optimal performance in the worst-case scenario. A policy together with a disruption scenario realization yields an itinerary. The output of our model can be the policy itself or the realized itinerary in the for any disruption scenario.

Literature review

For an introduction to robust optimization see the book by Ben-Tal *et al* (2009) and the references therein. In particular, the survey by Bertsimas *et al* (2007) provides some useful perspective. In Stojkovic *et al*. (2002), an extension of PERT/CPM models is proposed for solving *Day of Operations Scheduling* problems. However, this model only allows small ground delays as recourse decisions and it can only handle *small* disruptions. Smith *et al*. (2004) present an incremental optimization approach for the Barrel Master's problem at Air Mobility Command (AMC). Wilkins *et al*. (2008) propose a decision support system for AMC flight managers that identifies disruptions that require corrective actions and offers suggestions for dynamic rescheduling of missions. Finally, Wu *et al*. (2009) survey the various simulation and optimization approaches that one may take for airlift problems.

There are many variants of shortest path problems with cost uncertainties, for example, Montemanni & Gambardella (2005) give a method for solving shortest path problems, which have interval uncertainties on edge costs. The applicability of their approach for airlift problems could be somewhat limited.

The most important reference in the remainder of this article is a recent article by Bertsekas (2015) who develop algorithms for robust optimization under uncertainty. We adapt and simplify their general approach to our problem. Also relevant to this discussion is the work on shortest path games by Yu (2014).

An extended state-space graph

We shall assume that we are given a planned itinerary that serves a set of demands. In other words, we are given a sequence of stops that must be traversed in a specified order. We are also given the minimum length of on ground time that an itinerary must spend at each stop, the maximum duty period $\mathcal{M}$ for the crew, and a time horizon of $\mathcal{N}$ periods. We consider stochastic disruptions that make certain locations unavailable for landings and takeoffs for a length of time.

We consider discretized time and duty hours and model the problem on an extended location-time graph. This graph represents the state space associated with our scheduling problem. We incorporate crew duty cycles in the description of our graph.

Nodes: A node denoted (l, t, d, a) where $l \in \{0, \dots, \mathcal{L}\}$, is the set of locations, $t \in \{0, \dots, \mathcal{N}\}$, $d \in \{0, \dots, \mathcal{M}\}$, and $a \in \{\text{arr}, \text{dep}\}$specifies not just location and time but also the duty cycle – the number of hours that the crew on board has already worked. We do not allow nodes where the duty cycle exceeds $\mathcal{M}$. For each time and elapsed duty cycle (t, d), every geographical location l in the set $\mathcal{L}$ gets represented twice in the set of nodes, once as an arrival location (l, t, d, arr) and once as a departure location (l, t, d, dep). In addition there is a super source S and a super sink D that are connected to all departure nodes from $l = 0$, and all arrival nodes at $l = \mathcal{L}$ respectively.

Edges: There are several different classes of edges in this graph. We enumerate these classes below:

1. Edges that go from an arrival node (l, t, d, arr) to departure nodes $(l, t + \Delta, \max\{d + \Delta, \mathcal{M}\}, \text{dep})$. Depending on the nature of the stop (e.g., loading, unloading, crew change, refueling, etc), the aircraft needs to remain on ground for a minimum length of time; this determines the allowable values of Δ. Such edges correspond to a 'regular stop' at a given location i.e., a stop without any crew rest or crew swap.
2. Edges that go from an arrival node (l, t, d, arr) to departure nodes $(l, t + \Delta, 0, \text{dep})$. These edges correspond to a 'crew swap' since the duty hours are reset to zero as a fresh crew is being assigned.
3. Edges that go from a departure node (l, t, d, dep) to an arrival node $(l', t + T(l, l + 1), d + T(l, l + 1), \text{arr})$. These edges represent the travel between two locations and $T(l, l + 1)$ is the travel time between the locations. If $d + T(l, l + 1) > \mathcal{M}$ then there is no such edge because the crew would exceed its duty cycle in mid air.

Edge weights: Weights are assigned to edges to account for flying costs, delay penalties, and crew costs.

1. Each travel edge carries a weight equal to the flying cost.
2. Each edge representing a regular stop carries a zero weight.
3. An edge corresponding to a crew swap carries the cost of a crew swap at this location.
4. The edges that connect the last location in the stop sequence to the super sink D carry a weight equal to the late delivery penalty or non-delivery penalty (if applicable). The delivery penalty is calculated based on a target delivery time and is a function of the deviation from the target time.

Disruptions and Recourses: In keeping with our discretization of the problem, a disruption at a location may have a finite (usually small) number of possible realizations. Each such realization is characterized by three pieces of information: the location, a start time t_{start}, and an end time $t_{\text{end}} > t_{\text{start}}$. The effect of a realized disruption is to prevent landings and/or takeoffs at the affected location for all times between t_{start} and t_{end}, both inclusive. A node (l, t, d, a) is affected by a realization $(l, t_{\text{start}}, t_{\text{end}})$ of a disruption if $t_{\text{start}} < t < t_{\text{end}}$. Each realization affects the planned itinerary differentially. In particular, ground delays are introduced in the schedule for departures, and air delays and diversions are introduced for disrupted arrivals. This effect of a realization is modeled in our approach by associating the disrupted node (l, t, d, a) with another node (l, t', d', a) that serves as its recourse. Note that $t' > t$ so that the recourse node is always later in time and moreover is not in turn disrupted by the same event. Thus, for each potentially disrupted node i we define a set of recourse nodes R_i associated with all realizations of the disruption.

Robust itinerary generation

The problem of finding a robust itinerary given multiple disruptive events, with finite discrete realizations, is equivalent to finding a path from S to D that minimizes the maximum cost over all realizations.

Policies: A policy chooses exactly one outgoing edge from each vertex. In other words, a policy tells us the next action to take when we find ourselves in a particular state.

Itinerary: An itinerary is a sequence of nodes $\{(l_i, t_i, d_i, a_i)\}_{i=0}^{n}$ such that there is an edge between any two consecutive nodes in the sequence. In other words, an itinerary is a path from the super source to the super sink in our directed graph.

Given a scenario, i.e. a choice of realizations for each disruptive event, a policy always yields a path that is feasible in that scenario. A policy when coupled with a particular choice of disruption realizations yields a path in space-time. The cost of a path is the sum of the edge costs. The goal of this paper is to minimize the cost of the worst-case path that can result from a policy. This worst-case path corresponds to a particular scenario, which could be selected by a malicious opponent. Unlike the classical robust itinerary, which is feasible in every scenario, this worst-case path is not necessarily always feasible. If a different realization should occur, then the worst-case itinerary would be disrupted as well. However, the optimal policy is then able to show the best recourse to take after the subsequent disruption.

In the figure below, we show some of these features on a section of the graph when in-air delays are disallowed. In this picture we have two locations, one of which is affected by a disruptive event (indicated by the boxed nodes). The first two "rows" are respectively the arrival and departure nodes at this location. The third row shows he arrival nodes at the subsequent location. Time is discrete and in this picture it runs from t=0 to t=5. The elapsed duty cycle is also discretized and it can take values between d=0 and d=3. For each time t, the nodes corresponding to different duty cycles are shown diagonally for ease of visualization. The dashed edges show regular stops and flights. Each solid line connects an affected node to its recovery or recourse node. Note that these solid lines are not edges in the graph. Because in-air delays are disallowed, every affected arrival node gets a node at the end of time horizon as its recourse to make landing at such affected nodes infeasible. The solid edges that go from nodes with d=2 or d=3 to a node with d=0, correspond to a ground delay with a crew change. Otherwise they correspond to a ground delay without a crew change.

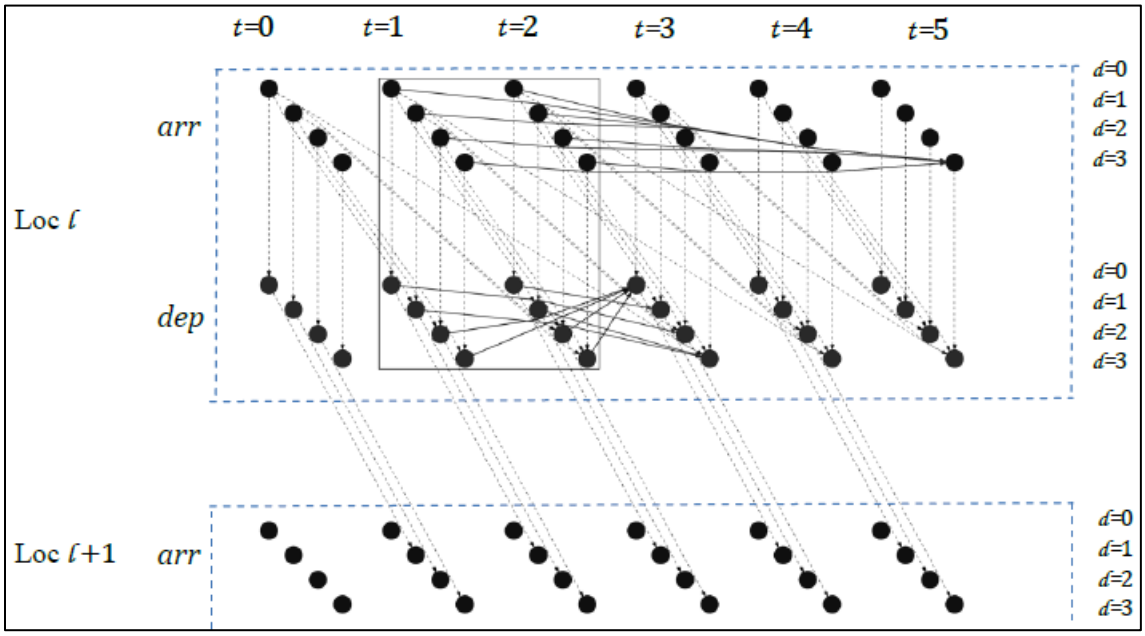

Figure 1. Extended State-space Graph with Recourses

Algorithm

The labeling algorithm proceeds as follows. Each node gets two labels, denoted l and L. We iterate over the nodes in anti-lexicographical order. As we do this, for each node i, we update its own label L_i and the labels l_j for each of its parents.

Initialization: Set $L_i = 0$, $l_i = \infty$, and $Succ(j) = \emptyset$ for all i

Iterative Step: For each node i in anti-lexicographical order

- Set $L_i = \max_{k \in R_i} \{l_k\}$
- For each j such that (j,i) is an edge update $l_j = \min\{l_j, c_{ji} + L_i\}$, where c_{ji} is cost of the edge (j,i). If $l_j = c_{ji} + L_i$, then $Succ(j) = i$.

Termination: At termination the label of the super source node L_S gives the length of the robust itinerary. The robust itinerary can be traced by starting from super source S and tracing through the $Succ$ labels to the super sink D. The complete policy is obtained by looking at $Succ(i)$ for all i.

Results

The following table shows the scaled penalty in the worst-case for the policy generated by the algorithm. The scenarios for the table above were generated for a problem with 6 stops. We had two disruption events at locations 2 and 5. In the 4-scenario case there were two different realizations for each of the disruptions in terms of start and end times. The 9- and 25-scenario cases were similarly generated. As indicated earlier in this paper, we assume that all take offs and landings are disallowed when a location is affected by a disruption. Other parameters of the problem were held fixed. We solved the above problems using a time discretization of 1 hour, 30 minutes, and 15 minutes respectively.

Table 1. Performance of Robust Policy. (No disruptions = 1.00.)

	1 hour discretization	*30 minute discretization*	*15 minute discretization*
4 (2x2) scenarios	1.084	1.041	1.030
9 (3x3) scenarios	1.094	1.062	1.041
25 (5x5) scenarios	1.105	1.073	1.052

Each figure in the table is the ratio of the worst-case cost of the policy to the cost of the undisrupted shortest path. Thus, in the case of 4 scenarios and a discretization of 1 hour, the robust itinerary has an 8% higher cost than the shortest path. The results confirm the intuition that as we add more scenarios, certain policies no longer remain cost effective and thus the robust solution worsens. Furthermore, finer time discretization gives the algorithm more choices, in time spent on ground for example, and it can then find a policy with a lower cost. This is reflected in the 5% lower cost of the resulting solution for 15-minute discretization as compared to the 1-hour case. The improved performance comes at the cost of longer compute times. Because the time discretization leads to an increase in nodes due to both shorter time intervals and more discretized duty cycles, the solution time increase from under one second for 1-hour to approximately 10 minutes for 15-minute discretization. Note that the algorithm finds an optimal policy, which means it finds recourse action for every combination of realizations of the disruptive events.

Conclusions

This paper focuses on disruption management for airlift operations, a problem of considerable practical interest real-world transportation. Our model and methodology robustly handle disruption uncertainties and because of an expanded decision space, our policies yield paths that are shorter than a single robust itinerary that avoids all disruptions. At the same time, we make no assumptions about the probability distributions on the sets of disruptive events. At the same time, a decision maker has the flexibility of deciding which disruptions to plan for. The algorithm presented here is polynomial time. Implementations of this algorithm in high-level programming languages such as Python or C++ are able to solve typical problems with a 15-minute discretization, 5-10 stops, and with two or three choices for disruptive events at each stop, in just a few minutes on a modern laptop.

Acknowledgments—This work has been partially supported by grants from MITRE Corporation.

References

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski, (2011) Robust optimization. Princeton University Press.

Dimitris Bertsimas, David B. Brown, Constantine Caramanis (2011). Theory and applications of Robust Optimization. SIAM Review, Vol. 53, No. 3, pp. 464–501.

Goran Stojkovic, Francois Soumis, Jacques Desrosiers, Marius M. Solomon, (2002). An optimization model for a real-time flight scheduling problem. Transportation Research Part A: Policy and Practice, Vol. 36, No. 9, pp. 779–788.

S.F. Smith, M.A. Becker, L.A. Kramer, (2004). Continuous management of airlift and tanker resources: A constraint-based approach. Mathematical and Computer Modelling, 39(6-8), 581-598.

David E. Wilkins, Stephen F. Smith, Laurence A. Kramer, Thomas J. Lee, Timothy W. Rauenbusch, (2008). Airlift mission monitoring and dynamic rescheduling. Engineering Applications of Artificial Intelligence, 21(2), 141-155.

Tongqiang Tony Wu, Warren B. Powell, Alan Whisman, (2009). The optimizing-simulator. ACM Transactions on Modeling and Computer Simulation, 19(3), 1-31.

Montemanni, R., & Gambardella, L. M. (2005). The robust shortest path problem with interval data via Benders decomposition. 4OR, 3(4), 315-328.

Dimitri P. Bertsekas, (2015). Robust Shortest Path Planning and Semicontractive Dynamic Programming. Report LIDS – 2915, MIT.

Huizhen Yu (2014). Stochastic Shortest Path Games and Q-Learning. LIDS REPORT 2875, MIT.