Proceedings of the 2015 Winter Simulation Conference
L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.

ROBUST AND FLEXIBLE AIR MOBILITY SCHEDULING USING STOCHASTIC SIMULATION-BASED ASSESSMENT

Talib S. Hussain
Richard Shapiro

Information and Knowledge Technologies Unit
Raytheon BBN Technologies
10 Moulton Street
Cambridge, MA 02138, USA

Steve R. Sommer
Nicole Ogden
John Dea

Llamasoft, Inc.
720 Olive Street
Suite 755
St. Louis, MO 63101, USA

John Collins
Julia Baum
Adan E. Vela
Allison Chang

MIT Lincoln Laboratory
Massachusetts Institute of Technology
244 Wood Street
Lexington, MA 02420, USA

ABSTRACT

We introduce an effort to create prototype capabilities to enable the Analysis of Mobility Platform (AMP) to produce airlift schedules for the Agile Transportation for the 21st Century (AT21) program at the United States Transportation Command (USTRANSCOM) that are more robust and flexible to real world changes. AMP currently uses a deterministic simulation-based process to produce schedules, effectively assuming that execution occurs under expected conditions. We have designed robustness and flexibility heuristics that generate different candidate schedules, and a stochastic simulation that varies departure delays using a probabilistic model based on real-world Global Decision Support System (GDSS) data. Through stochastic simulated executions of candidate schedules and several robustness/flexibility measures, including a *schedule content comparison* metric, our approach seeks the candidate schedule that best balances solution quality with robustness/flexibility. We present our heuristics, stochastic model and measures, and summarize our initial findings and next steps for robust and flexible AT21 scheduling.

1 INTRODUCTION

The Analysis of Mobility Platform (AMP) has been a Department of Defense (DoD) model of record for programmatic analysis for the past two decades. AMP models all modes of travel (air, land, and sea) to execute the delivery of cargo specified in Time-Phased Force and Deployment Data (TPFDDs). This is essentially a large, multi-faceted Vehicle Routing Problem with Pickup and Delivery and Time Windows. AMP is based on a planning and scheduling algorithm that operates over time as part of a deterministic discrete-event simulation. Traditionally, AMP has been used for large-scale simulations up to several years in duration in order to explore programmatic analyses, such as the impact of different asset acquisitions. A recent program at the United States Transportation Command (USTRANSCOM) called Agile Transportation for the 21st Century (AT21) has been focused on introducing more optimization technology into day-to-day operations at USTRANSCOM and its components. To aid in developing a process to solve the strategic airlift scheduling problem, USTRANSCOM decided to modify AMP to model this operational problem using the current state of the world and to schedule thirty to forty-five days of real cargo requirements that will move beginning in 10 to 14 days out. During an earlier initial phase of effort (Sommer et al. 2014), a novel schedule optimization capability was developed and deployed that used column generation and set covering approach to generate globally cost-optimal missions for a given problem. This new capability demonstrated a very large reduction in cost and lateness when compared against the existing AMP heuristics for an acceptable increase in planning time by the combined optimizer and AMP process. However, the AT21 use case requires more than schedules that just minimize cost and lateness. It also requires schedules that can effectively accommodate the typical issues that arise during real-world execution of a complex schedule – either by building in enough *robustness* to enable accommodation of minor changes to a mission's execution with no (or limited) impact on other missions, or by building in enough *flexibility* to enable operational users to repair more significant changes to a mission's schedule with minimal impact on other missions.

We introduce work underway in a second phase of effort that seeks to prototype new schedule optimization capabilities to enable AMP to produce schedules that are quantifiably more robust/flexible. To accomplish this, we have designed and performed early prototyping of three key capabilities:

1. *Candidate Generation:* The generation of multiple candidate schedules that can potentially vary in their robustness/flexibility in order to support a search-based algorithm. A key method is to apply heuristics to incorporate robustness and/or flexibility properties into schedules, such as incorporating different types of slack (Ahmadbeygi, Cohn, and Lapp 2009; Carey 1994; Lan, Clarke, and Barnhart 2006) into the schedules or using network isolation methods (Rosenberger, Johnson, and Nemhauser 2004) to minimize ripple effects.
2. *Stochastic Simulation:* The stochastic variation of different operational delay values in the AMP discrete-event simulation in a manner that models delay variations in real-world Global Decision Support System (GDSS) data. Within the simulation, a schedule is adaptively executed using the schedule's robustness/flexibility properties while obeying all constraints enforced by AMP.
3. *Robustness/Flexibility Assessment:* The quantitative assessment of the robustness/flexibility of a given candidate schedule based on how it executes in stochastic simulations. This enables the comparison of different candidate schedules to determine their relative robustness/flexibility.

We introduce the methods we have designed and prototyped for each of these capabilities, as well as our approach to integrating these capabilities into a single prototype system that generates more robust/flexible schedules. The effort, which began in 2014 and will complete in 2016, continues to refine our approach to achieving prototype robust and flexible AT21 scheduling using AMP.

2 APPROACH

Figure 1 illustrates our approach for creating robust/flexible schedules. It is a search-based method in which a variety of candidate schedules C_i are first generated for a given scenario using an enhanced version of the AMP Schedule Optimizer (SO). A candidate schedule represents a moderate-fidelity representation of how cargo should move within the system on each day of the scenario, and does not capture all the detailed timing of a complete schedule. Each of these candidates is expected to show some differences in how it executes in the face of real-world events, and the goal of the approach is essentially to test out each of those schedules in a number of stochastic simulation runs to gain an empirical estimate of each candidate's robustness/flexibility.

The first step in assessing each candidate is to establish its baseline scheduling performance. This is accomplished by executing the AMP detailed scheduler and discrete-event simulation using its traditional deterministic approach, which essentially provides typical, average values for execution variables, such as the amount of time taken to service, load or unload a given type of aircraft at a given port. In creating this baseline schedule (S_i), all timing details are worked out to the minute level, and all constraints of a legal schedule are enforced, such as not exceeding available service resources, crew duty limits or port congestion limits. That baseline schedule also establishes a baseline performance against the scheduling objective functions, such as the amount of cargo successfully moved, total cost and average lateness.

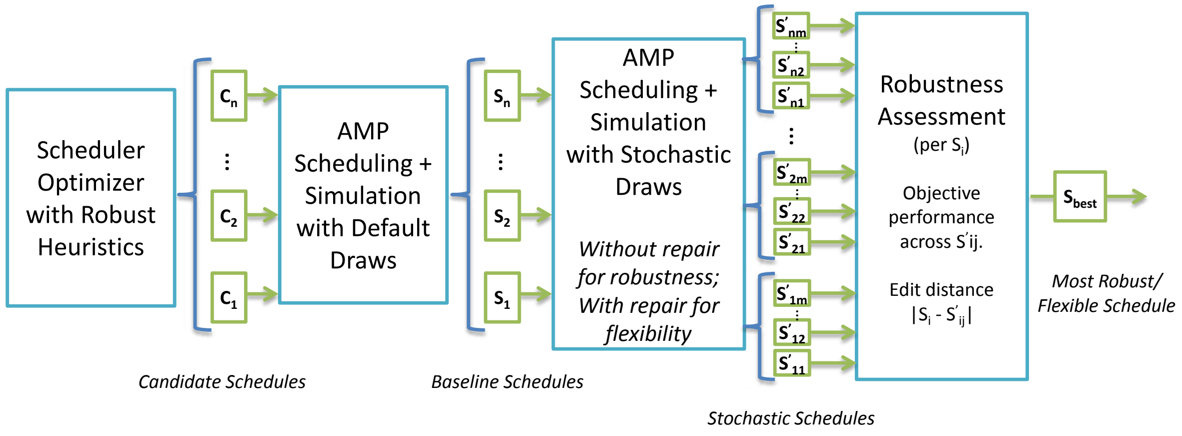

Scheduler Optimizer with Robust Heuristics
C_n
C_2
C_1
AMP Scheduling + Simulation with Default Draws
S_n
S_2
S_1
AMP Scheduling + Simulation with Stochastic Draws
Without repair for robustness; With repair for flexibility
S'_nm
S'_n2
S'_n1
S'_2m
S'_22
S'_21
S'_1m
S'_12
S'_11
Robustness Assessment
(per S_i)
Objective performance across S'ij.
Edit distance |S_i - S'_ij|
S_best
Most Robust/ Flexible Schedule
Candidate Schedules
Baseline Schedules
Stochastic Schedules

Figure 1: End-to-end approach for producing robust/flexible schedules.

Once a baseline is established, each candidate schedule is then executed multiple times using an enhanced AMP simulation that stochastically varies the values of certain execution variables. The result of each stochastic run is a fully-instantiated schedule, and for each original candidate schedule, a set of instantiated schedules is obtained (S'_{ij}). To measure only robustness, the scheduler is run in a mode where it makes no repairs to the schedule when random variations result in a conflict. To measure flexibility and robustness, the scheduler performs minor repairs of the schedule when needed.

Finally, to obtain an assessment of the overall robustness/flexibility of the original candidate schedule, each instantiated schedule is compared to the baseline schedule, both in terms of its difference in performance, as well as in terms of how different the specific schedules are. A particular measure, termed the *schedule content comparison metric*, quantifies differences between the content of baseline and instantiated schedules (denoted $| S_i – S'_{ij} |$) by assigning "cost" values to the individual changes made during schedule repair. The statistical pattern of performance differences and schedule content comparison measures across the entire set of instantiated schedule is used to compute a single overall measure of the robustness/flexibility of the original candidate schedule. After assessing each candidate, the highest scoring candidate (S_{best}) is chosen as the most robust/flexible solution.

3 CANDIDATE GENERATION

Several methods are used to generate candidate schedules in order to support enhanced robustness and flexibility, including capturing opportunistic candidates along the way to optimal, inserting slack along a route to absorb system delay when needed, and isolating subnetworks to minimize cascading disruptions.

3.1 Opportunistic Candidates

Within AMP, the Schedule Optimizer (SO) uses a series of Mixed Integer Program (MIP) solves to generate medium-fidelity schedules that are globally optimized against several performance criteria

(Sommer et al. 2014) using the classical technique of branch and bound. Within branch and bound, interim integer solutions are found along the way to optimal (it is these solutions that are compared to the current bound to determine optimality) as part of the solution process. One approach we have used to generate different candidate schedules is to have the solver export some of these interim solutions.

There are a few drawbacks to relying upon capturing these solutions along the way. Primarily, there is no guarantee that the incremental solutions will be of sufficient quality. For example, it is not uncommon for the MIP to find a solution that is 20% from optimal and then jump to a solution that is 2% from optimal that stops the search. Further, there is no particular guarantee that a given set of these solutions along the way will vary meaningfully from the final solution in terms of their robustness. For example, it is not uncommon for the MIP to find many highly similar solutions as it approaches optimal. Because of these limitations, additional heuristic approaches to candidate generation are also used.

3.2 Slack Insertion

Slack insertion is the purposeful addition of less-than-efficient delays in a schedule in order to improve the realized execution of the schedule which is subject to unexpected delays due to weather, unexpected closings, over-capacitated airports, and so on. It is also a technique to reduce churn later in the scheduling process where new requirements or changes to existing missions are necessary due to real-world constraints as execution time approaches. Slack can provide flexibility for planners to adjust to the inevitable changes that occur as a schedule is taken from initial planning to eventual execution.

The motivation for including slack in the planning stage rather than relying solely on ad-hoc solutions is that the SO creates efficiency in the entire system. Planning for such delays allows the optimizer to organize the schedule often much better than only accounting for the true delays ad-hoc, even if the planned schedule has unnecessary delays. Another motivation for including slack is that the SO only plans on the day level. The simulation creates the instantaneous plan from that schedule which can result in even more delay if the instantaneous airport capacity used is greater than expected. As these delays propagate, the simulation must deviate more and more from the optimized plan, as the constraints become more violated – likely causing cascading delay.

Figure 2 illustrates the delay propagating through a system that was planned without slack. In this schedule, the delay impacts the arrival and departure times at all future stops on the route. These delays will also impact all other traffic at the airports along the route.

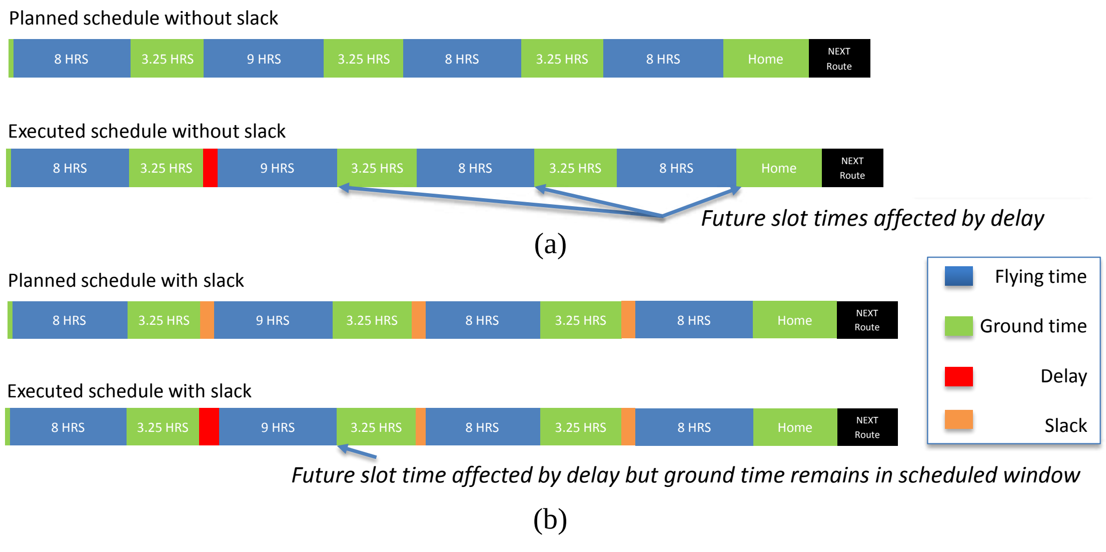

Planned schedule without slack
8 HRS
3.25 HRS
9 HRS
3.25 HRS
8 HRS
3.25 HRS
8 HRS
Home
NEXT Route
Executed schedule without slack
Future slot times affected by delay
(a)
Planned schedule with slack
Executed schedule with slack
Future slot time affected by delay but ground time remains in scheduled window
(b)
Flying time
Ground time
Delay
Slack

Figure 3: (a) Schedule without slack; (b) Schedule with slack.

In our enhanced SO, the schedules produced add slack along the route. Figure 3 illustrates the same notional stop sequence as in Figure 2, but with slack added at each ground stop. A delay at a particular node impacts the departure time at the first stop along the route as well as the arrival time at the following stop. However, because of the slack inserted the disruption ends there - the departure time at the second stop remains within the scheduled window and all subsequent stops are unaffected.

The resulting SO schedules will necessarily be less effective/efficient with added slack because of additional lateness; however, the detailed end solutions from the stochastic simulations may be more effective. A key goal of our enhanced SO is to determine how much slack to add without making the solution less efficient overall than necessary. The optimal solution without slack is often too tight. However, adding too much slack or adding it naively will cause unnecessary lateness and increases in cost. We have created several different slack heuristics in order to explore different approaches (based on different intuitions) to determining where to insert slack and how much slack to insert, including:

1. *Even slack:* Our simplest heuristic adds the same amount of slack to every stop along the route. This approach captures the intuition that many types of delays (such as weather or equipment failure) can occur at any time and as such are equally likely across the route.
2. *End-of-route delay:* While even slack can be effective at accommodating delays wherever they may occur, another intuition is that the need for slack may occur more toward the end of the route (e.g., as network congestion is encountered or as execution-delay effects accumulate). Our second heuristic adds no slack in the beginning of the route, but gradually increases the amount of slack in the middle of the route, and adds a larger amount of slack for stops toward the end of the route.
3. *Congested-forward slack:* The intuition behind the congested-forward slack heuristic is that unexpected delays, and therefore propagated delays, tend to occur after congested locations on a route. The heuristic has several implementations. One is to only add slack to stops on a route that occur after a congested location. A second (called congested-forward-scaled) is to increase the amount of slack at stops after each successive congested location the route visits. A third (called congested-forward-tapered) reduces the amount of slack after consecutive non-congested stops. A fourth implementation uses the scaled and tapered policies together.
4. *Slack budget:* All the previous heuristics have the SO determine specific slack amounts at specific nodes along the route. However, a different intuition is that this imparts a certain amount of rigidity to the medium-fidelity SO schedule that may not be appropriate given the extra knowledge that is available to the detailed AMP scheduler. Our fourth key approach is to identify a certain total amount of slack to insert in an SO schedule, but then allow the detailed AMP scheduler to vary where along the route to place that slack when computing the baseline schedule. This allows, for example, instantaneous estimates of congestion computed in AMP to influence where the slack is placed. This slack budget approach is implemented an additional constraint in the MIP, and as an enhancement to the AMP detailed scheduler.

These heuristics have been prototyped to demonstrate feasibility. However, while they provide approaches to varying the amount of slack along the route, work continues on determining the actual amounts of slack to apply. Our plan is to leverage historical data on average delays at different locations, as well as to empirically test out the impact of different slack values on our robustness/flexibility assessments, in order to find an appropriate balance between robustness and inefficiency.

3.3 Network Isolation

Our third approach to generating candidate schedules is to isolate the locations into distinct networks, assigning a set of aircraft to fly only to/through a subset of locations, rather than allowing any aircraft to go to any base. This approach is much like an airline with a hub-and-spoke network assigning aircraft to only a single hub and having as few aircraft as possible travel between different hubs.

Sometimes an entire airport becomes either unavailable, such as due to weather or an insurgent attack, or heavily congested. The more aircraft scheduled to use that airport, the more routes are impacted by the closure/delay, and the worse the final result becomes. By isolating bases into separate networks, we reduce the impact to the overall schedule. Only the few aircraft using that base are affected, with no effect on the rest of the schedule.

There is a trade-off involved here. By enforcing this segmentation, we disallow possible route combinations. As a result, the overall schedule will not be the “optimal” deterministic one. However, we expect that when real-world events interfere with the planned schedule, this approach will perform better than one optimized for a perfect world.

Our network isolation method begins with the set of pickup-drop off pairs and builds a connectivity graph (matrix) showing which bases are linked to each other. We make a connection between two bases if there is a requirement for a pickup at one to be delivered to the other. This graph may not be fully connected. When it isn’t, the separate fully-connected portions make up our initial sub-networks. Figure 4 shows an example of this. The different colors highlight the “natural” sub-networks from the requirements set. Depending on the requirements, one or more of these “natural” sub-networks may contain an undesirably large number of bases. When that happens, we can subdivide them further, having only one or two bases overlapping and being assigned to more than one sub-network.

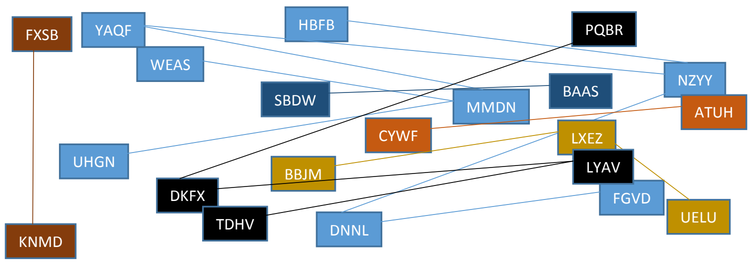


Figure 4: Requirements graph.

The next step is to assign a home base and an en-route base to each sub-network. A pair of bases is selected based on the average route distance for each requirement, with some internal logic to balance the overall utilization of each home base and the capacity needed to satisfy all the requirements. There may be more sub-networks than available home bases, so it is possible for more than one sub-network to be assigned to the same home base. However, what is not allowed is for an en-route base to be paired to more than one home base.

The third phase, counter-intuitively, is to consolidate the sub-networks. The rationale is this: The more segmented the networks are, the fewer routes can be generated, and the further from optimal the final solution will be. Hence, by performing some reconsolidation, we can restore some of the route generation flexibility and get a better solution while still keeping the most important aspects of the network isolation. This consolidation can be one of four types:

1. Do no consolidation at all. Maintain complete segregation among the sub-networks.
2. Sub-networks which originate from the same home base can use any en-route base paired with that home base.
3. Sub-networks which originate from the same home base are consolidated into a single larger sub-network. The difference between this option and the previous one is that a single route can satisfy requirements from distinct initial sub-networks.
4. An aircraft assigned to one home base can use any base and fulfill any requirement, so long as it does not use an en-route base assigned to a different home base. This option gives the most flexibility in route selection while keeping the important isolation of segregating the transoceanic refueling stops.

The final phase involves generating the routes used by the MIP within the Schedule Optimizer. During route generation, the network isolation restricts which missions can be joined into a single route. When prepending the aircraft home base to a requirement mission, the only route generated is the one from the home base assigned to that requirement's sub-network. This restriction results in an immediate, massive reduction in the number of routes generated. When appending pickup-drop off pairs to the growing route, the network isolation again restricts which routes can be generated, in several ways. First, regardless of the network isolation option, new missions are limited to those that can be reached via the restricted set of en-route bases. Second, the specific network isolation option limits which potential pickup-drop off pairs can be appended. Finally, the return trip home is limited to certain en-route stops.

We have prototyped the network isolation capability to demonstrate its feasibility, and plan to empirically explore the degree to which different sub-network consolidation approaches impact the robustness and flexibility of our solutions.

4 STOCHASTIC SIMULATION

AMP currently contains a deterministic simulation that executes scheduled missions over time in tandem with the scheduling process. Specifically, during the run of AMP, the simulation clock moves forward one day at a time. Scheduling decisions are made each day and those decisions, as well as any earlier ones, are simulated as appropriate for that day. To create an effective stochastic simulation capability, we designed an efficient enhancement of the current AMP scheduling approach, where a single AMP run becomes a single stochastic run. In this "Stochastic AMP" approach, the core scheduling process of AMP is used to apply and enforce the myriad of constraints built into that process. Rather than using only the deterministic values of the current code, however, the scheduler randomly determines certain key variables for each mission, such as the ground time spent by a particular aircraft at a particular airport on a particular day. These random values are used within the AMP algorithm to schedule all the specific details of the mission, which are then simulated as usual by AMP.

To generate and keep track of the random values, we integrated a Drools rule engine with AMP. The Drools rules specify the types of random factors to apply at particular points in the code, and enable the simulation to draw from the appropriate probability density functions for those factors. This approach applies an important consistency rule to ensure that the stochastic simulation is meaningful: The AMP constraint scheduler is currently designed to perform a search over multiple possible ways to schedule a mission. To ensure that the search is not simply choosing the best set of random variables, all random draws for that mission are recorded. If a particular draw has been made for that mission at a particular port on that particular scheduling day, then the exact same value is used for that mission on that day at that port every time the search explores that option (e.g., when considering different routes that overlap at that port, or when exploring alternative scheduling decisions for the same route)..

This approach enables the entire AMP scheduling process itself to introduce stochasticity while producing a valid scheduling solution. An important additional requirement of this approach is the ability for AMP to accept a partial or fully defined schedule and then seek to match it as best as possible during the stochastic scheduling process. A single "Stochastic AMP" run will accept a prior schedule, schedule/simulate it one day at a time while determining stochastic events and computing stochastic values (particularly certain ground times and en-route times) based on a Drools fact base. The fact base is a separate component that contains the stochastic modeling rules to apply and the current stochastic state across all missions, ports and days. The final schedule produced is a complete schedule where the specific scheduling decisions (e.g., how long spent at each port) reflect the stochastic events.

4.1 Modeling Underlying Factors

At the simplest level, a stochastic simulation merely requires that certain elements in the simulation be randomly varied. In a scheduling problem as complex as the AMP problem, however, there are a number of elements that are meaningfully inter-related, and hence should be meaningfully inter-related in a stochastic simulation. Within our approach, certain random variations are based, where appropriate, on underlying stochastic factors rather than solely as independent random events.

The easiest-to-understand example of an underlying stochastic factor is weather. Say the simulation "rolled the dice" and determined that there was a delay – due to weather – of a certain length for a given mission at a given airport on a given day. Consider now the next mission at the same airport on the same day. A naïve stochastic simulation might randomly roll the dice with no regard for the previous roll and come up with no delay due to weather. While the overall effect of the rolls might capture the random variability at the large scale based on the underlying probability density functions, that naïve simulator would not provide meaningful insights into the effect of weather on a planned schedule. A more sophisticated stochastic simulator, by contrast, would instead "roll the dice" to determine if there was weather at that port on that day, and then randomly determine the impact of that weather on all missions flying through that port on that day. This approach enables the analyst to more easily see the impact of weather-related effects on a schedule, as well as provide a more meaningful assessment of how robust a schedule was to realistic weather events. However, it does not require a complete model of weather around the world to be created – just a consistent treatment of related stochastic draws at shared locations.

4.2 Stochastic Model

We have developed a stochastic simulation that uses the underlying-factor approach based on a stochastic model derived from real-world data. There are two high-level types of variability that typically occur in the real world – changes in requirements and delays during mission execution. Data on the former may be obtained from the Joint Operation Planning and Execution System (JOPES), but understanding and incorporating this type of variability into a stochastic simulation is a significantly complex problem. Much more tractable to understand and model is the data on mission delays, which is available from the Global Decision Support System (GDSS). Hence, we focused on modeling delays.

A GDSS data set was collected containing a large number (~260) of departure delay codes – each specifying a different reason for a delay (e.g., equipment issues, weather, crew issues) – for over 170,000 missions. For every stop at which a mission is delayed more than 15 minutes in its departure beyond what was initially planned, GDSS in principle contains a delay code and delay amount corresponding to the primary reason for that delay. A detailed analysis and logical grouping of this data was performed to produce an ontology (see Table 1) for interpreting the diverse delay codes into an actionable framework.

Table 1: Mission delay ontology.

Components	Attributes
Number of Aircraft /Missions Affected	One / Many
Relation to Aircraft/Mission	Internal / Environmental
Disruption Process	Probabilistic / Deterministic
Causality	Direct / Indirect
Service Delay (or Shift)	Yes/Yes (shift)/ No
Maintenance Delay	Yes/ No
Location Delay Occurred	Local/ En-Route/ Destination

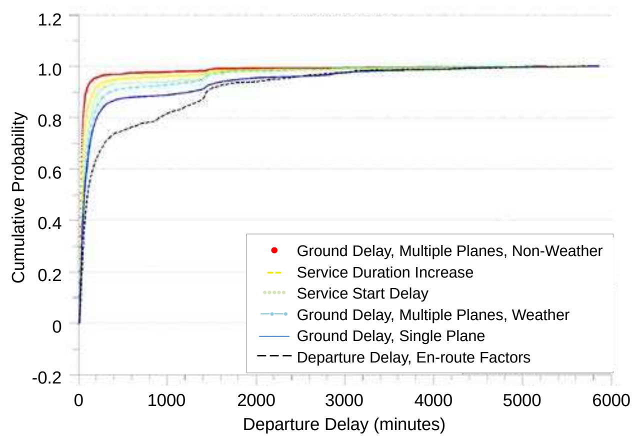
An important aspect of this ontology is that it enables multiple delay codes with the same classifications to be grouped together to determine a single common probability density function. In this way, specific delay codes can be abstracted out of the stochastic simulation. Instead, common types of random variations can be modeled instead. This approach provides significant flexibility in both the data analysis and the stochastic modeling method. Another important aspect of this ontology is that it can be used to filter out data that will not be of specific use in the stochastic simulation approach we are using. In our current model, three key types of delays codes are not included:

- "Deterministic" delays effects are ignored since they are already computed by AMP. For instance, the AMP automatically computes the crew rest times so that they obey constraints. So, if there are certain random delays in a mission that result in the need for the crew to rest, then AMP will already automatically insert that rest.
- Delays that occur at "Destination" locations are ignored since the current AMP scheduling algorithm sequentially makes all decisions at the current node independently of the next node.
- "Indirect" delay effects are ignored since they are already computed within AMP (e.g., a delay caused by another aircraft's delay will already propagate in AMP).

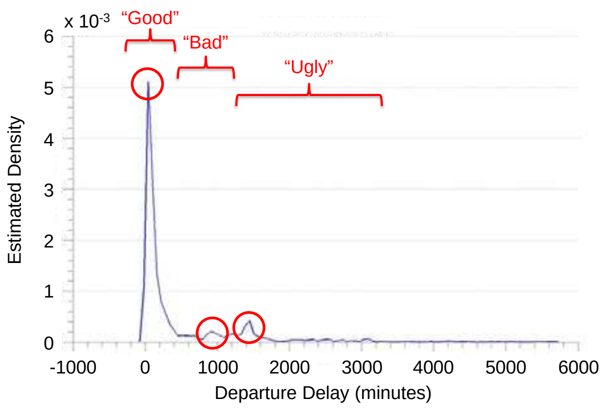
Further, we used only data for the first delayed sorties on each mission to avoid double-counting delays and excluded sorties for which the delay codes were ambiguously defined. This ontology was used to compute initial probability density functions (PDFs) representing the stochastic variability for six different delay variables:

1. The delay in the start of service for a plane at a port.
2. The increase in service duration for a plane at a port.
3. The total ground delay for a plane at a port.
4. The total ground delay for multiple planes at a port due to weather.
5. The total ground delay for multiple planes at a port due to non-weather factors.
6. The departure delay for a plane flying between two ports due to en-route non-weather factors.

The resulting PDFs for each delay type are shown in Figure 5a. These PDFs were generated using only the 50 most frequently-occurring delay codes in the data set. Each PDF is drawn as a cumulative distribution function (CDF). All of these six distributions are statistically different distributions, which is a valuable result since it means that it is meaningful to distinguish these variables from each other in our model. Together, these distributions account for 62% of the non-propagated delay (matching our criteria above) in the data set, and form a strong basis for our initial stochastic model of delay. As we continue our work in our second year, the remaining codes will be classified and added to our model.



(a)



(b)

Figure 5: (a) Cumulative distribution functions for the six stochastic delay variables within our model; (b) Breaking up a PDF into portions that can be drawn from separately.

Several of the PDFs have a multi-modal nature (see Figure 5b). Our model uses these modes to further break up the overall distribution into portions that can be randomly drawn from in an independent manner (e.g., the "Good", "Bad", "Ugly" portions shown in Figure 5b). This is of particular use for delay factors that affect many aircraft since it enables us to ensure that different aircraft experience an underlying delay issue of similar magnitude (e.g., if the weather at a port on a given day is “Bad”, all affected aircraft at that port experience delays drawn from the “Bad” portion of the distribution).

5 ROBUSTNESS/FLEXIBILITY ASSESSMENT

The ultimate goal of our assessment approach is to provide a value that indicates the “goodness” of the original baseline schedule so that it can be meaningfully compared against other baseline schedules and the “best” one chosen. “Good” and “best” will ultimately reflect a tradeoff between the quality of the solutions on the primary performance criteria (e.g., lateness, cost) and ability of a baseline schedule to appropriately (i.e., robustly and flexibly) handle realistic stochastic events. We use several measures for assessing the quality, robustness and flexibility of a given baseline schedule.

Measuring solution quality is straightforward using statistical measures of key performance variables (e.g., timeliness of delivery, amount of cargo moved, and cost) across the set of stochastic schedules. However, measuring the ability of a schedule to handle stochastic events may be done in several ways:

- *Pre/post differences:* Computing the difference between the baseline (“pre” stochasticity) schedule and the instantiated schedules (“post” stochasticity) to get an estimate of how much has to change in the former to produce the latter. The schedule content comparison measure (see next section) can be viewed as an estimate of “the number of phone calls” that would be needed by human schedulers in real life to handle corresponding delay events.
- *Stochastic dial:* Determining how much stochasticity can be applied to the baseline schedule before solution quality deteriorates below an acceptable level. We can vary this “stochastic dial” by increasing or decreasing the number and magnitude of the variations allowed. A baseline schedule that produces better quality stochastic solutions under high stochasticity is deemed more robust than one that produces worse ones.
- *Repairability:* Estimating the scale of repairs that must be made to the baseline schedule in order to produce a high quality solution. While performing the stochastic scheduling, multiple alternative scheduling choices may be considered (e.g., different ways to handle an issue that arises where the schedule needs some type of repair). Given a particular repair algorithm, a baseline schedule that tends to require more, extensive repairs is deemed less flexible than one that requires fewer, simpler repairs. The key to this measure is deriving an estimate of repair complexity from the actions of the repair algorithm itself. The measure is effectively a direct measure of “the number of phone calls” needed to handle the random events, and is related in principle to the schedule content comparison metric.

We have primarily focused on creating the schedule content comparison measure and demonstrating its initial feasibility. As we proceed, we will explore the correlation between the repair measure and the schedule content comparison measure.

5.1 Schedule Content Comparison

A key concept in the assessment of robustness is the quantification of differences between baseline schedules and the stochastic schedules resulting from simulation of disruptions and replanning. That is, a robust schedule is one that can preserve quality (lateness, resource utilization efficiency, etc.) while requiring a minimal number of changes from the planned baseline. While metrics for schedule quality are widely known and employed in airlift planning, metrics for schedule content difference are less commonly used.

Techniques for quantifying the difference between a pair of entities often assign abstract “cost” values to operations by which one can be “edited” to resemble the other. The technique is most often used to quantify differences between character strings (Levenshtein 1966) or graphs (Gao et al. 2010), but the notion can be made to apply to airlift schedules as well. That is, the difference between a baseline and stochastic schedule can be seen as resulting from a series of three types of "edit" operations: (1) modifications to mission parameters, (2) deletion of planned missions, and (3) insertion of new missions. A logically consistent cost scheme would assign lower costs to less impactful changes, for example a slight mission delay relative to a complete mission cancellation. Specific cost values for operations affecting scheduled missions are best defined in consultation with subject matter experts, to assure that the resulting measures of schedule differences capture intuition about the severity or impact of different types of mission changes.

Given a scheme for assigning costs to mission changes, differences in schedule content can be associated with the "edit distance" arising from the sequence of changes by which the baseline schedule is realized within the simulation. So, for example, a schedule with slack inserted using the techniques of Section 3.2 tends to absorb unexpected delays, resulting in fewer mission failures related to asset unavailability. Since fewer missions would need to be modified or cancelled, the content comparison measure would tend to be lower, capturing the benefit of that particular robust scheduling approach.

6 CONCLUSIONS

We have presented an effort to develop prototype capabilities to enable the Analysis of Mobility Platform (AMP) to support the generation of schedules that are more robust and flexible to real-world execution events. Our approach leverages stochastic simulation as the basis for assessing the robustness/flexibility of schedules, and applies several different methods for generating potentially-robust candidate schedules, as well as several methods for assessing that robustness/flexibility using the stochastic simulation results. In our first year of effort, we have demonstrated the feasibility of the component capabilities of our process. Our goal for our second year is to build out those capabilities and integrate them in a prototype system that produces reliable assessments of robustness and flexibility to support AT21 scheduling.

ACKNOWLEDGMENTS

We would like to thank our teammates Rob Seater at Lincoln Laboratory and Francisco Jauffred, Atikhun Unahalekhaka and Chris Caplice at Massachusetts Institute of Technology for their design contributions. We would like to thank Peter Matthews, Thomas Whieldon, Patrick McLeod and Theresa Baynes at the USTRANSCOM Joint Distribution Process Analysis Center for their support of this work. This work was sponsored by USTRANSCOM under USTRANSCOM Contract No. HTC711-14-D-D001-0005 and Air Force Contract No. FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

REFERENCES

Ahmadbeygi, S., A. Cohn, and M. Lapp. 2009. "Decreasing Airline Delay Propagation by Re-allocating Scheduled Slack." *IIE Transactions* 42:478–489.

Carey M. 1994. "Reliability of Interconnected Scheduled Services." *European Journal of Operational Research* 79, no. 1:51–72.

Gao, X., B. Xiao, D. Tao, and X. Li. 2010. "A Survey of Graph Edit Distance." *Pattern Analysis and Applications* 13(1): 113–129.

Lan, S., J. P. Clarke, and C. Barnhart. 2006. "Planning for Robust Airline Operations: Optimizing Aircraft Routings and Flight Departure Times to Minimize Passenger Disruptions." *Transportation Science* 40(1): 15–28.

Levenshtein, V. I. 1966. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals." *Soviet Physics Doklady* 10:707.

Rosenberger, J., E. L. Johnson, and G. L. Nemhauser. 2004. "A Robust Fleet-Assignment Model with Hub Isolation and Short Cycles." *Transportation Science* 38:357–368.

Sommer, S., J. Judd, T. S. Hussain, and G. Grindey. 2014. "End To End Modeling (ETEM): Scheduling Optimization." *82nd Military Operations Research Society (MORS) Symposium,* Presentation #591.

AUTHOR BIOGRAPHIES

TALIB S. HUSSAIN is a Senior Scientist in the Information and Knowledge Technologies business unit at Raytheon BBN Technologies. He holds a PhD in Computer Science from Queen's University at Kingston. His research interests include evolutionary computation and optimization methods for transportation logistics and game-based training for individual and team decision-making. His email address is thussain@bbn.com.

STEVE R. SOMMER is a Director of Defense Research and Development in Public Sector Solutions at LLamasoft Inc. He holds a BS in Applied Mathematics from Southern Illinois University Edwardsville. His research has included optimization methods for airlift scheduling in transportation logistics. His email is steve.sommer@llamasoft.com.

JOHN COLLINS is a Technical Staff member at MIT Lincoln Laboratory. He holds a PhD in geography from the Center for Remote Sensing at Boston University. His research has included modeling and simulation, data mining, and analytic algorithm development for intelligence and transportation logistics applications. His email address is jcollins@ll.mit.edu.

JULIA BAUM is an Assistant Technical Staff member at MIT Lincoln Laboratory. She holds a BS in Mathematical Sciences from Worcester Polytechnic Institute. Her research is focused on mathematical optimization and data analysis with applications in national defense and air transportation. Her email is julia.baum@ll.mit.edu.

RICHARD SHAPIRO is a Senior Scientist at BBN Technologies. He has degrees in Math and Computer Science, and over 35 years of experience in a range of languages, platforms and development environments. His primary interest is software engineering quality: maintainability, extensibility, transferability, readability, generality, and efficiency. His email address is rshapiro@bbn.com.

NICOLE OGDEN is an Operations Research Scientist in Public Sector Solutions at LLamasoft, Inc. She holds a master's degree in Operations Research and Industrial Engineering from the University of Texas at Austin. Her email address is nicole.ogden@llamasoft.com.

JOHN R. DEA is a Senior Operations Research Scientist (Public Sector Solutions) at LLamasoft, Inc., and a former Assistant Professor of Mathematics at the Air Force Institute of Technology. He also serves part-time as an Adjunct Instructor of Mathematics at Liberty University and Indiana Wesleyan University. He holds a PhD in Applied Mathematics from the Naval Postgraduate School. His research interests include numerical modeling of wave propagation in large and unbounded domains. His email address is john.dea@llamasoft.com.

ADAN E. VELA is Technical Staff in the Air Traffic Control Division of MIT Lincoln Laboratory. He obtained his PhD in Mechanical Engineering from the Georgia Institute of Technology. His research interests include the application of control theory and optimization towards air transportation systems, particularly related to human-factors and decision support systems. His email address is adan.vela@ll.mit.edu.

ALLISON CHANG is a Technical Staff member at MIT Lincoln Laboratory. She holds a PhD in Operations Research from the Massachusetts Institute of Technology. Her research focuses on the application of optimization and statistical learning models in support of national defense. Her email address is aachang@ll.mit.edu.