Project Proposal: Simulation Engine for Evaluating GDP Planning Methods

Alex Estes
Advisor: Dr. Michael Ball

October 19, 2014

1 Background

Bad weather can cause the capacity of an airport to accept to decrease. If flights were allowed to depart normally, then many flights may have to wait at the airport to land. This can cause an unmanageable and unsafe situation. In order to prevent this, the Federal Aviation Administration will delay flights at their origin in anticipation of bad weather, so that delays are experienced on the ground instead of in the air. This is called a Ground Delay Program (GDP). After delays have been assigned in a GDP, the airlines will make changes to their schedules. The airlines are allowed to swap the scheduled arrival times of any of their flights. For example, consider the case that an airline has a flight with original ETA of 8:00 and a flight with original ETA of 8:30, and these flights are delayed to 9:00 and 9:30 respectively. The airline may decide that the 8:30 flight is more important, and will choose to have the 8:30 flight arrive at 9:00 and the 8:00 flight arrive at 9:30.

Currently, the FAA allocates delays in a GDP using a method called Ration-By-Schedule (RBS). In RBS, flights are given assignment priority based on their scheduled Estimated Arrival Time (ETA). Those with earliest ETAs are the first flights allowed to arrive at the airport. Many alternatives to RBS have been proposed. It is not easy to determine how effective a proposed GDP planning method is. The authors who propose these methods usually evaluate their performance either by performing a simple simulation, or by reporting the value of some objective function that the method uses. For example, an author may create GDP plans by solving an integer program (IP) with expected total delay as the objective function. That author might report the expected total delay of the optimal solution to the IP. Most of these methods involve simplifying assumptions for the sake of computational tractability. Due to these assumptions, the reported objective value will be inaccurate. Furthermore, to the best of my knowledge, none of the GDP planning literature takes into account the response of airlines when evaluating the performance of the GDP planning method.

2 Project Goal

The goal of this project is to more accurately evaluate GDP planning methods and to incorporate airline response into the evaluation of GDP planning methods.

3 Approach

My approach will be to create a program which simulates a GDP. The program will take a list of flights and airspace parameters, and create objects which represent the flights and airport under consideration. The program will allocate delays using whichever GDP planning method is being evaluated. Then, airline responses and aircraft departures and arrivals will be simulated. Finally, the program will produce metrics which may be used to evaluate the effectiveness of the GDP planning method.
3.1 Program Organization

The framework will be organized into four parts:

1. Input parser
2. State update module
3. GDP planning module
4. Airline response module

The input parser will take input data and create a state object. This object is a representation of the airspace, and is described more completely below. The state update module will take the state object and advance it in time. The GDP planning module will allocate delays according to some specified GDP planning method. This will change the scheduled departure and arrival times of some of the flights. The airline response module will produce the reaction of the airlines to the delays issued by the GDP planning module, which will again alter the scheduled departure and arrival times of flights. The flow of the data through the program is described in the diagram below:

Figure 1: Proposed Data Flow

The dotted edges occur once, the dashed edges occur infrequently and the solid edge occurs frequently. This process is also described in high-level pseudocode below:
**Algorithm 1** Program Organization

**INPUT:** As described above
1: Call Input Parser to create state representation
2: for t = start time to end time do
3:   Call state update module to advance simulation in time
4:   if GDP planning condition is met then
5:     Call GDP planning module to allocate delays according to specified GDP planning method
6:   end if
7:   if Airport response planning condition is met then
8:     Call airport response module
9:   end if
10: end for
11: return computed metrics

### 3.2 Inputs

The required data fields that will need to be provided by the user are listed below.

**Flight information:**
- Flight number
- Airline
- Origin
- Number of Passengers
- Original scheduled departure time
- Current scheduled departure time
- Original scheduled estimated time of arrival (ETA)
- Current scheduled ETA

This flight information describes all of the flights which are relevant to the GDP, describes their status when the simulation begins.

**Information for GDP planning module:**
- Which GDP planning method to use
- Tunable parameters for method if applicable

The GDP planning methods that will be implemented are specified later in this document. Some methods have parameters which need to be specified.

**A set of destination capacity scenarios. For each one:**
- GDP start time
- GDP end time
- Times at which the GDP will be rescheduled
- Times at which the airline response will be run
- Time resolution of capacity scenarios
• Capacity in each time interval
• Probability

In reality, capacity at the destination airport is dependent on the weather conditions at that time. In order to reduce the complexity of the simulation, we will describe the capacity of the airport directly. A capacity scenario describes the arrival capacity of the airport at each time, and also when the GDP will start and end. A scenario should also describe the start time and end time of a GDP, as well as the times at which the GDP planning method will be invoked and the times at which the airline response mechanism will be called. These times may be the same in all scenarios. Finally, a probability will be assigned for each scenario. This is the probability that this scenario occurs in a single complete run of the simulation.

Other parameters:
• The time at which the simulation starts
• The time at which the simulation ends
• Time resolution of simulation
• Parameters describing distribution of actual departure times
• Parameters describing distribution of actual arrival times
• Parameters describing distribution of critical times

Here, the critical times are the earliest time when a crew member needs to arrive at the airport in order to be able to make their connection. These, along with the actual departure times and actual arrival times will be randomly generated, with a distribution described by the user. I plan on implementing several possible distributions, including an exponential distribution, binomial distribution, and Poisson distribution.

3.3 State Object

After these are input, a state object will be created. The state object will be a representation of the state of the air space as is relevant to making GDP planning decisions. It will include all of the inputs listed above, and will also include the following:

Flight Information:
• Actual departure time
• Actual arrival time
• Critical time

The actual departure time will be randomly generated from the input current scheduled arrival time. The actual arrival time will be randomly generated when the plane takes off, and may be delayed if the flight has to wait to land at the destination.

Delay Metrics:
• Total flight delay
• Total passenger delay
• Total planned ground delay
• Total air delays
• Total delay from random events
The program will keep track of these metrics as the simulation progresses. These metrics measure different types of delay in the system. The total flight delay is the sum of delay for each flight. The total passenger delay sums the delay of each flight multiplied by the number of passenger on that flight. The planned ground delay for a flight is the difference between the flight’s scheduled departure time in the GDP and the flight’s original scheduled departure time. Total planned ground delay is the sum of these delays. Finally, total delay from random events is the amount of delay that is caused by the random perturbations of schedule which the simulation will produce. Once the simulation is completed, it will be these metrics that we may use to compare the effectiveness of various GDP planning methods.

**Scenario Information:**

- The actual scenario
- A scenario tree

From the possible scenarios, a single actual realized scenario will be randomly selected according to the probability distribution that has been specified. The idea of the scenario tree is that the depth of the tree represents the time which has passed, and a node in the tree represents the set of scenarios which are identical up to some time $t$. For example, consider 5 capacity scenarios:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>3:00-3:59</th>
<th>4:00-4:59</th>
<th>5:00-5:59</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>25</td>
<td>30</td>
</tr>
</tbody>
</table>

Then we may form the tree, shown below in Figure 2. As the simulation progress, we will track the position

![Figure 2: Example Scenario Tree](image)

of the realized scenario in this tree. Some GDP planning methods require us to input scenarios. When we do this, we do not wish to include scenarios which we already know are not the realized scenario. So, we only want to include scenarios which are identical to the realized scenario up to the current time, which is the node of the tree which we occupy at the current time. Then the scenario tree provides the inputs for these GDPs. One GDP planning method requires a scenario tree as an input. For this planning method, we would then input the subtree rooted at thes node which we currently occupy.
4 Scientific Computing Algorithms

4.1 State Update module

The state update module performs relatively inexpensive computations. These operations include determining which flights take off, which flights lands, updating scheduled departure times and ETAs, and calculating performance metrics. The steps taken are described below in Algorithm 1.

Algorithm 2 State Update

INPUT: a state object, as described in appendix one

1: for all flights $f$ whose actual departure time is the current time period do  
2:   Handle flights trying to depart
3:   Let $a$ be the origin airport of flight $f$.
4:   Flight $f$ departs.
5:   Set an actual time of arrival with some randomness
6:   Update metrics
7: end for

8: for all flights $f$ whose actual time of arrival is the current time period do  
9:   Handle flights trying to land
10:   if enough time has passed since last arrival at destination then
11:      Flight $f$ lands.
12:      Set a time with some randomness until next flight may land
13:      Calculate total delay and update delay metrics
14:   else
15:      Delay $f$’s current departure ETA by one time period
16:      Update delay metrics
17: end if
18: end for

4.2 GDP Planning Algorithms

I will implement several GDP planning modules. I have divided these into three categories: rationing algorithms, assignment-based integer programs (IPs) and network-based IPs. In all of these methods, we are not allowed to delay flights which have already taken off or flights which are exempt from the GDP. So, the first step of any of these methods will be to generate a flight list which consists solely of those non-exempt flights which have not taken off. The exempt and airborne flights are incorporated into the decision in different manners depending on the solution method, which will be discussed in the presentation of these methods.

4.2.1 Rationing Algorithms

The rationing algorithms I will implement are Ration-By-Schedule (RBS) [2], Ration-By-Distance (RBD), and Equity-Constrained Ration-By-Distance (RBD-E) [3]. In each of these, we need to convert the arrival capacity into a list of time slots, and then we ‘ration’ the slots among aircraft according to some priority rule. We incorporate the exempt and airborne flights by assigning these flight slots first. The algorithm for this initial assignment is given below. In RBS, we assign slots with priority to the flights which have the earliest original times of arrival. See Algorithm 2 below.
Algorithm 3 RBS

**INPUT:** A list of time slots and a list of flights.
1: Sort flights by increasing original ETA
2: for all \( f \), flights not airborne and non-exempt do
3: Set the scheduled ETA of \( f \) to the earliest unoccupied slot which is later than its original ETA
4: end for

RBS is widely considered a fair way to allocate delay to flights. However, other methods may be reduce delay. In RBD, priority is determined first by distance with further flights getting priority. Flights with equal distances receive priority based on original ETA. See Algorithm 3 below.

Algorithm 4 RBD

**INPUT:** A list of time slots and a list of flights.
1: Sort flights by decreasing distance and then by increasing original ETA
2: for all \( f \), flights not airborne and non-exempt do
3: Set the scheduled ETA of \( f \) to the earliest unoccupied slot which is later than its original ETA
4: end for

Under some conditions, RBD can be shown to produce the delay-minimizing slot allocation. On the other hand, RBD can be highly inequitable, since short-distance carriers will be receive disproportional amounts of delay. In order to balance delay and equity, a slightly more complicated algorithm called E-RBD algorithm was produced. For each flight, we assign a temporary slot using RBS. We then try to give the long distance flights earlier slots by making trades with shorter distances flights. However, there are restrictions on the trades that we can make. We cannot move a flight to a slot before its original ETA and we cannot move any flight to a slot where it would receive more than a prespecified amount of delay. See algorithm 4 below.

Algorithm 5 E-RBD

**INPUT:** A list of time slots and a list of flights and a maximum allowed delay \( D \)
1: Sort flights by increasing original ETA
2: for all \( f \), remaining flight do
3: Set the scheduled ETA of \( f \) to the earliest unoccupied slot which is later than its original ETA, mark this slot as temporarily occupied
4: end for
5: Sort flights by decreasing distance and then by increasing original ETA
6: for all \( f \), not airborne, non-exempt do
7: Find the earliest temporarily occupied slot that is later than the original ETA of \( f \) such that swapping time slots with the occupant would not delay the occupant more than the maximum allowed delay \( D \).
   Swap \( f \) into this slot, and mark the slot as permanently occupied.
8: end for

4.2.2 Integer Programs

In all of the integer programs that I will implement for the GDP planning module, the time period of the GDP will be discretized into a set of \( T \) of time periods. An additional time period \( T + 1 \) is usually added with high capacity, so that leftover flights may all be accommodated in this time period. All IPs will be solved with the use of the Gurobi commercial solver.

Mukherjee and Hansen proposed a stochastic IP which is based on an assignment problem [6]. The IP formulation requires us to form the scenario describe in section 3.3: State Object. Let \( Q_t \) be the set of nodes of those tree at some distance \( t \). Let \( N_t = |Q_t| \) and let \( \Omega_{i,t} = \{S_1^{i,t}, S_2^{i,t}, ..., S_{n_i,t}^{i,t}\} \) be the \( i \)th set of \( Q_t \) where
\( n_i^t = |\Omega_{i,t}| \). That is,
\[
Q_t = \{\Omega_{1,t}, ..., \Omega_{n,t}\} = \left\{ \left\{ \Omega_{1i,t}^1, \Omega_{1i,t}^1, ..., \Omega_{ni,t}^1 \right\}, ..., \left\{ \Omega_{1i,t}^N, \Omega_{1i,t}^N, ..., \Omega_{ni,t}^N \right\} \right\}
\]

where each \( \Omega_{i,t} \) is a maximal set of scenarios which are identical up to time \( t \). The IP proposed by Mukherjee and Hansen is described below.

**Mukherjee-Hansen Dynamic Model**

**Parameters:**
- \( \Theta \): the set of capacity scenarios
- \( M_q^t \): the capacity in scenario \( q \) at time \( t \) for \( q \in \Theta \)
- \( P_q \): the probability of scenario \( q \in \Theta \)
- \( \Phi \): the set of flights
- \( T \): the number of GDP time periods
- \( \text{Arr}_f \): the earliest time period in which \( f \) may arrive for \( f \in \Phi \)
- \( \text{Dep}_f \): the earliest time period in which \( f \) may depart for \( f \in \Phi \)
- \( \lambda \): the ratio of cost of air delay to ground delay
- \( W_{\text{max}} \): the maximum number of flights allowed to queue at the destination
- \( Q_t, N_t, \Omega_{i,t}, n_{i,t} \): described above

**Decision Variables:**
- \( X_{q,f,t}^q \): Binary variable, 1 if flight \( f \) is assigned to arrive at time \( t \) in scenario \( q \).
- \( Y_{q,f,y}^q \): Binary variable, 1 if flight \( f \) is assigned to depart at time \( t \) in scenario \( q \)
- \( W_q^t \): the number of flights at in the arrival queue the end of time period \( t \)

**Objective Function:**
\[
\min \sum_{q \in \Theta} \left( \sum_{f \in \Phi} \sum_{t=\text{Arr}_f}^{T+1} (t - \text{Arr}_f) X_{q,f,t}^q + \lambda \sum_{t=1}^{T} W_q^t \right)
\]

**Constraints:**
\[
Y_{q,f,t}^q = X_{q,f,t+\text{Arr}_f-\text{Dep}_f}^q \\
\sum_{t=\text{Arr}_f}^{T} X_{q,f,t}^q = 1 \quad \forall f \in \Phi, q \in \Theta \\
W_q^t - W_q^{t-1} + \sum_{f \in \Phi} X_{q,f,t}^{1} \leq M_q^t \quad \forall t \in \{1, ..., T+1\}, q \in \Phi \\
W_q^1 = W_q^{T+1} = 0 \quad \forall q \in \Theta \\
Y_{q,f,t}^{S_1} = Y_{q,f,t}^{S_2} = ... = Y_{q,f,t}^{S_{ni,t}} \quad t \in \{1, ..., T+1\}, i \in \{1, ..., N_t\}, n_{i,t} \geq 2 \\
X_{q,f,t}^q \in \{0, 1\}, Y_{q,f,t}^1 \in \{0, 1\}, W_q^t \geq 0
\]
The next two IP models that I will implement do not directly assign departure and arrival times to flights. Instead, they decide how many flights should have their arrivals delayed. One of the previously discussed rationing algorithms can be used to convert the solution of these IPs into an actual flight schedule. The Hofkin model published by Ball, Hoffman, Odoni and Rifkin [1] is given by:

**Hofkin Model**

Parameters:
- \( T \): the number of time periods in the discretization of the GDP
- \( S_t \): The number of flights with earliest arrival times in time period \( t \)
- \( Q \): Set of capacity scenarios
- \( D_{t,q} \): The arrival capacity of the airport in time periods \( t \) in scenario \( q \)
- \( p_q \): the probability that capacity scenario \( q \) occurs
- \( a_t \): cost of air holding at time \( t \)
- \( g_t \): cost of ground holding at time \( t \)

Variables:
- \( x_t \): number of flights with arrival time in period \( t \) which are assigned to arrive in time period \( t \)
- \( y_t \): number of flights with arrival time in period \( t \) who are ground delayed at least until period \( t + 1 \)
- \( z_{t,q} \): number of flights which arrival in time period \( t \) who are air delayed at least until time period \( t + 1 \) in scenario \( q \)

Objective function:

\[
\min \sum_{t=1}^{T+1} \left( g_t y_t + \sum_{q=1}^{Q} p_q r_t z_{t,q} \right)
\]

Constraints:
- \( x_t + y_t - y_{t-1} = S_t \) \quad \forall t \in \{1, ..., T + 1\}
- \( z_{t-1,q} + x_t - z_{t,q} \leq D_{t,q} \) \quad \forall t \in \{1, ..., T + 1\}, q \in Q
- \( y_0 = y_{T+1} = 0 \)
- \( z_{0,q} = z_{T+1,q} = 0 \) \quad \forall q \in Q
- \( x_t, y_t \geq 0 \) \quad \forall t \in \{1, ..., T + 1\}
- \( z_{t,q} \geq 0 \) \quad \forall t \in \{1, ..., T + 1\}, q \in Q

The final IP GDP planning model that I will implement is a model proposed by Richetta and Odoni which is a similar model to the Hofkin model [4] with the difference that the IP is formulated as an assignment problem instead of a flow problem. Their model uses almost exactly the same parameters, with the exception that instead of a constant ground delay cost per time period, we have a possibly non-constant cost function which describes how much a ground delay of \( n \) time periods costs. Then, the IP is given below.

**Richetta-Odoni model**

Parameters:
- \( g(n) \): the cost of ground delaying a flight \( n \) time periods
- all others: as in Hofkin model
Variables:

\( x_{t,s} \) : number of flights with scheduled arrival time in period \( t \) which are assigned to arrive in time period \( s \)
\( z_{t,q} \) : number of flights which held in air at destination at end of period \( t \) in scenario \( q \)

Objective function:

\[
\min \sum_{t=1}^{T} \sum_{s=t}^{T+1} G(s-t)x_{t,s} + \sum_{q=1}^{Q} \sum_{t=1}^{T} p_q r_t a_t z_{t,q}
\]

Constraints:

\[
\begin{align*}
\sum_{s=t}^{T+1} x_{t,s} &= S_t & \forall t = \{1, \ldots, T\} \\
\sum_{t=1}^{s} x_{t,s} + z_{t-1,q} - z_{t,q} &\leq D_{t,q} & \forall s \in \{1, \ldots, T+1\}, q \in Q \\
z_{0,q} &= z_{T+1,q} = 0 & \forall q \in Q \\
x_{t,s}, z_{t,q} &\geq 0 & \forall t \in \{1, \ldots, T+1\}, q \in Q
\end{align*}
\]

4.3 Airline Response Algorithm

After delays have been allocated in a GDP, airlines are allowed to reschedule their flights by swapping the scheduled arrival times of any of their flights. This will also be modeled with an IP. Luo and Yu proposed an assignment problem that minimizes the total passenger delays with delay past critical times weighted higher. [5]. As with GDP planning models, airborne flights may not have their schedules altered. However, airlines may change the arrival times of exempt flights.

**Luo-Yu Airline Response model**

Parameters:

- \( \Phi \) : the set of non-airborne flights for the airline
- \( S \) : the set of arrival times that are available for the airline
- \( \text{Arr}_{f} \) : the earliest arrival time slot of \( f \) may arrive for \( f \in \Phi \)
- \( \text{Crit}_{f} \) : the critical time slot of \( f \)
- \( \mu \) : the extra cost of delay past critical time
- \( n_f \) : the number of passengers on flight \( f \)

Decision Variables:

\( X_{f,s} \) : Binary variable, 1 if flight \( f \) is assigned to arrive in slot \( s \).

Objective Function:

\[
\min \sum_{f \in \Phi} \sum_{\text{Arr}_f \leq s \leq \text{Crit}_f} n_f (s - \text{Arr}_f) X_{f,s} + \mu \sum_{\text{Crit}_f \leq s} X_{f,s}
\]
Constraints:

\[
\sum_{s=Arr_f}^S X_{f,s} = 1 \quad \forall f \in \Phi
\]

\[
\sum_{f \in \Phi} X_{f,s} = 1 \quad \forall s \in \{1, ..., S\}
\]

\[
X_{f,s} \geq 0 \quad \forall f \in \Phi, s \in \{1, ..., S\}
\]

In this case, we may simply solve the LP relaxation of the IP, because this IP has only integer extreme points.

5 Implementation

I will implement this using Java. I will use the Gurobi API for Java in wherever a solution to an integer program is required in the GDP planning module and airline response module. In general, the problem of solving an integer program is NP-Hard. Researchers have already performed computational tests with the integer programs that appear in this project and have deemed them computationally tractable. The most computationally intense of the IP models is that of Mukherjee and Hansen, and in their tests, they found that it took less than a minute to solve the IP. Each complete run of a simulation will only solve use the GDP planning modules a few times, so running the simulation once should not take very long. However, since there are stochastic elements, the simulation may need to be run many times to generate good estimates of the desired metrics. This could require a lot of computation. If this is a problem, parallelization could be considered.

For hardware, I will likely either use my personal laptop or the desktop that is in my office. The laptop has an Intel Core i5-4200M CPU dual-core 2.50 GHz processor with 8.00 GB of ram and the desktop has an Intel Core i7-4790 CPU with 3.60 GHz processor and 8.00 GB of ram.

I am likely to use the following two databases:

- FAA Advisories database: http://www.fly.faa.gov/adv/advAdvisoryForm.jsp
- BTS on-time performance database: http://www.transtats.bts.gov/Fields.asp?Table_ID=236

The FAA advisories database contains information about GDPS including start time, end time, and arrival rates. The BTS on-time performance contains information about the scheduled and actual departure time and arrival times for flights. These databases would be used to create realistic flight schedules for the simulation to run on. The aviation system performance metrics database contains records of performance metrics, which I would use for verification purposes. All of these databases are publicly accessible.

6 Validation Methods

Each module may be validated separately. I will validate the input parser by creating sample inputs, printing the objects that the input parser creates and then comparing the inputs with the printed objects. The non-stochastic elements of the state update module may be tested by setting the stochastic parameters such that there is no randomness, and then running scripted test cases which I know what the output should be. The stochastic elements of the state update model may be tested individually by running the simulation many times and comparing the generated results with the expected results. For example, I can choose the random parameters so that the arrival rate in some time period has a certain mean and variance, and then I may run the simulation many times and check that the observed arrival rate matches that distribution. The validation of the GDP planning module will vary by the method employed. For the rationing heuristics,
I can generate relatively small test cases that I can manually verify the correctness of. I can validate the IP modules by solving the IPs for some test problems, and then comparing the results I get to the results the authors get. For all of the IP modules, there are existing computational results. However, in most cases, the testing data is only partially described. Mukherjee and Hansen provide all of the parameters except for the flight schedules [6]. In this case, I will use flight schedules from the BTS in place of those that Mukherjee and Hansen used. The metrics produced will not be exactly the same since different inputs are used, but if the metrics produced by my implementation were very different from their reported metrics, this would indicate that my implementation is not correct. Kotnyek and Richetta have provided some results for both their IP model and the Hofkin model using inputs described completely in [8] and [7]. So, I can run these two models with these inputs and compare the results. Validating that the modules interact properly is simply a matter of examining whether the schedule alterations produced by the GDP planning module and airline response are incorporated correctly into the state object. This can be accomplished by manual examination of small or moderate sized test cases.

7 Test Problems for Verification

Performance metrics are known for any GDP that has been run in the past few years. These performance metrics are accessible from the aviation system performance metric database. The FAA uses the RBS method of allocating delay in a GDP. To test how well my simulation engine simulates actual air system behavior, I can run the simulation with the RBS method specified for the GDP planning module and with a flight schedule taken from actual BTS data and compare the output performance metrics to those that were observed on that day.

8 Results

I expect my project to produce a simulation framework that can more realistically evaluate the performance of a GDP planning method.

9 Concluding Remarks

The goal of the project will be to build a simulation framework to evaluate GDPs. This will be divided into an input parser, a state update module, a GDP planning module, and an airline response module. The input parser module will build an object which describes the relevant parts of the airspace. The state update module will simulate things such as arrivals and departures of flights. The GDP planning module will simulate FAA allocation of delay either by a rationing heuristic or through the solution of an IP. The airline response module simulates the reaction of airlines to the delays given by the FAA which will be found by solution of an IP. The input parser, state update module and rationing algorithms will be validated through the use of test cases which I will create, as will the integrated behavior of the simulation framework. The IP GDP planning methods and airline response module will be validated by running the modules on the test cases described by their authors if they are available, or approximations of those test cases if they are not available.

10 Timeline

March: Implement and validate IP-based planning modules.
April: Integrated testing, parallelization if necessary.
May: Prepare report
If time permits: Create graphic user interface for simulation.

11 Milestones

Milestone 1: Simulation framework with input parser and state update module
Milestone 2: Simulation framework with input parser, state update module, and RBS/RBD/E-RBD GDP planning modules
Milestone 3: Simulation framework with input parser, state update module, RBS/RBD/E-RBD GDP planning modules and airline response
Milestone 4: Completed simulation framework
Milestone 5a: Parallelized simulation framework if necessary and if time allows.
Milestone 5b: Graphic user interface if time allows.

12 Deliverables

The deliverables of the project are as follows:

- Weekly reports
- Project proposal
- Mid-semester report
- Mid-year status report
- Final report
- Complete simulation framework including input parser, state update module, GDP planning modules and airline response module
- Test cases for validation of each module
- Integrated test cases for validation of framework
References


