**Existing Scheduling System**

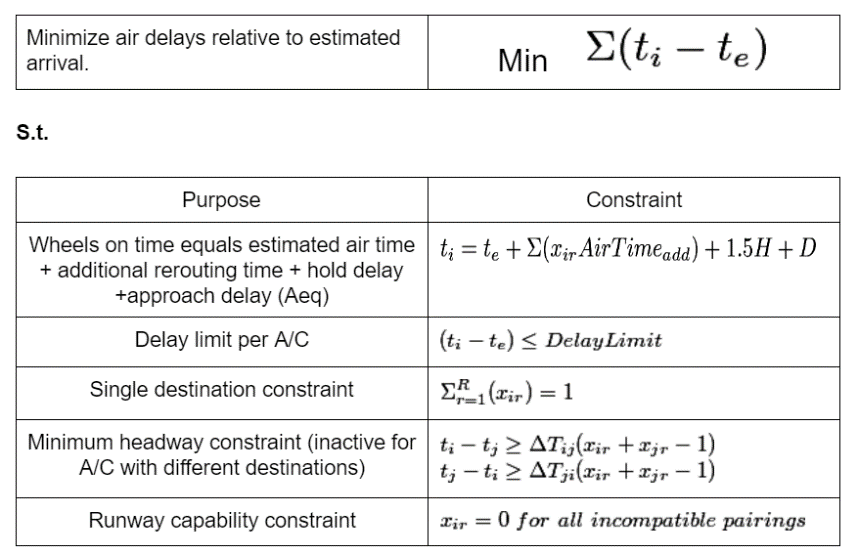
Currently, due to the Airline Deregulation act of 1978, the scheduling of flights is determined by market forces, airline decisions, and airport-determined rules for how slots can and cannot be used. In this system, airports offer slots to airlines, which set a maximum on airline service may be bounded by number of flights, or number of passengers. IATA’s Worldwide Slot Guidelines (WSG) also play a guiding role in determination of an airport’s slot policy [5].  
 In this framework, slots are distributed to airlines according to legal procedures and expected demand for each airline, and the airlines then schedule their flights in accordance with those allocations. Because of this, large and high capacity airlines are incentivized, making market entry for smaller airlines difficult. Generally, slots dictate how many gates are available as well so that slot policy, coupled with gate leases, makes for a very calcified and profit focused market at large, busy airports such as LAX. Furthermore, since both airlines and airports are competitors, there is often little to no communication between parties, leading to a schedule that may be optimal for each entity, but is not efficient globally, leading to conflicts and delays.  
 Though this practice leads to an inefficient and highly profit driven policy under the current paradigm, slots can be as restrictive as they need to be. This means airport slot policy could be a possible area of improvement or regulation to implement a more delay optimized schedule.

**Approach**

To approach a problem such as this with its high potential for political and economic impacts throughout the air traffic industry, our group determined that the primary issue in developing effective regulation was the information gap it would have to overcome. The decisions of passengers, airlines, and airports are influenced by a plethora of factors. Thus, to determine a schedule that could reduce delays, as much data as possible should be taken into account to accurately predict the effects of making such changes to avoid regulation inefficiencies due to a lack of information.  
 To address this problem, our project focuses on investigating and experimenting with different optimization techniques to demonstrate the merits of developing scheduling software that uses individual flight data to produce a delay-minimizing schedule and assess its impact on the air transportation industry of Los Angeles. Research into this topic throughout the semester yielded two promising methods: Mixed Integer Linear Programming (or MILP for short) and Dynamic Programming. Both have been used by previous work in the field of air traffic scheduling and can be applied to optimize a variety of factors [3] [2]. Computational methods are also widely used by the FAA to assist air traffic controllers, such as in the case of the Center TRACON Automation System (CTAS) used to schedule aircraft and maintain safe spacing [1]. Our problem expands upon this work with the added complexity of scheduling between multiple airports as options and minimizes delays where as others focus on maximizing aircraft separation.

**MILP**

At their core, MILPs are simply linear programs that utilize branch and bound to derive integer solutions and often apply the Big M Method to transform logical disjunctions into constraints that can be analyzed by a linear solver. Its benefits include its relative ease of implementation and flexibility in making changes or improvements later since the whole program depends mainly on changes to the constraint matrices. However, early results showed that MILPs of this variety are inherently time intensive, as its number of constraints and variables can easily balloon into the millions with less than a single day of arrivals to schedule.  
 Because of this, a set of assumptions had to be proposed and validated to make the problem feasible and the scope of the project had to be scaled back from arrivals and departures to just arrivals. Even with these assumptions though, the MILP suffers from an unpredictable time complexity that becomes infeasibly long in scenarios with difficult to locate optimums. Despite these flaws, the MILP remained the simplest to change and experiment with, and was selected as our method of choice to go ahead with further refinements.  
 Air traffic, with its many regulations and technical constraints, requires a properly posed problem to accurately represent real conditions. The below figure shows a summary of the problem’s components and how they were represented mathematically. Initially the problem was designed with the capability of determining the amount of holds and route delay that would be required to achieve an optimal distribution, providing further guidance for air traffic control decisions, but this added layer of detail was determined superfluous and a drain on resources. It was kept, but disabled to improve run time.   
 The other critical factors modeled by this problem are a delay limit to prevent any one aircraft from being forced to endure excessive delays and the objective function, which defines delay as the difference between scheduled arrival time and the earliest possible time that aircraft could arrive with an unimpeded approach.



|  |
| --- |
| MILP Problem Formulation |

In MATLAB, the following four steps detail the process for solving this problem given our available data.

1. Individual flight data is extracted from ASPM database[4]
2. Aircraft equipment ID and other databases are used to determine: ➤ weight category ➤ fuel consumption  
   ➤ point of origin ➤ original destination,   
   ➤ separation requirements ➤ time impacts of changing destination
3. For loops assemble the constraint arrays A, B, Aeq, and beq from the data calculated in Step 2
4. *Intlinprog* function uses the simplex algorithm and branch and bound to explore options and find a delay minimizing schedule

**Assumptions/Project Limitations**

The MILP makes a number of permissible assumptions about aircraft dynamics and regulations to enable faster computation. These assumptions include:

* Schedules repeat over the course of days or weeks.
* 7 Runways for arrivals (2 each at LAX and LGB).
* Aircraft arrive from 4 general directions, meaning additional flight time to each airport varies.
* Once at an airport, all aircraft merge into a single approach path

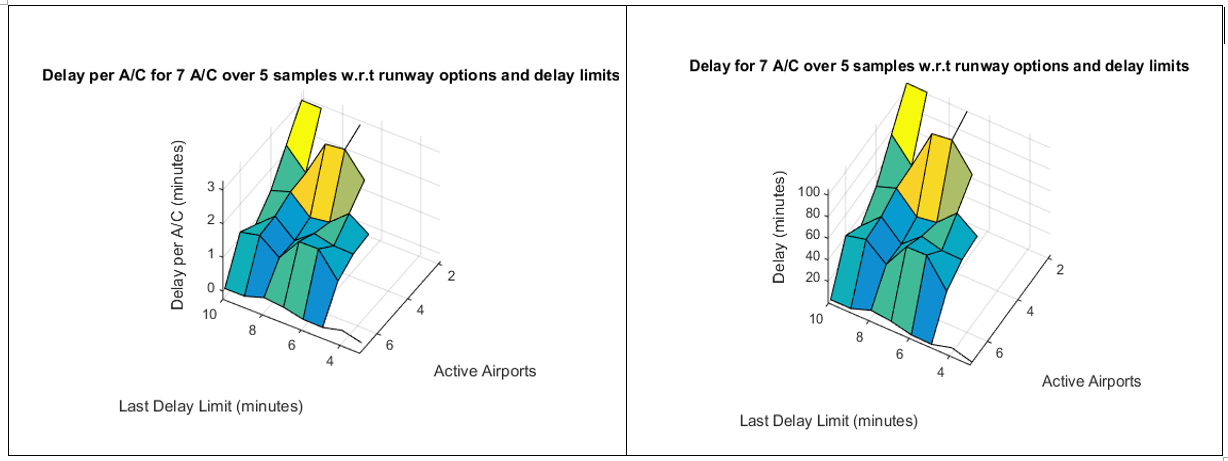
It also has a number of temporarily relaxed constraints. These were relaxed either to improve runtime or because of issues in finding, importing, and converting other data types to useable numeric data that could be read by a linear algebra solver. These relaxed constraints include:

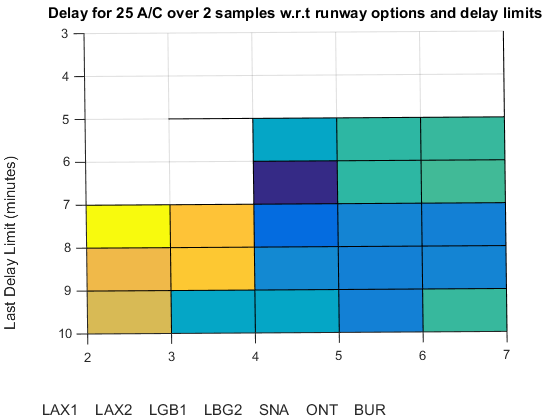
* Runway capabilities (based on weight class)
* Variable headway requirements (an average of 90 seconds used)
* Actual vs random origin direction

Beyond physical constraints, optimality conditions must also be taken into account, and tolerances on solutions and variables can be relaxed to improve time complexity in some large sample size scenarios.

**Bivariate Constraint Selection**

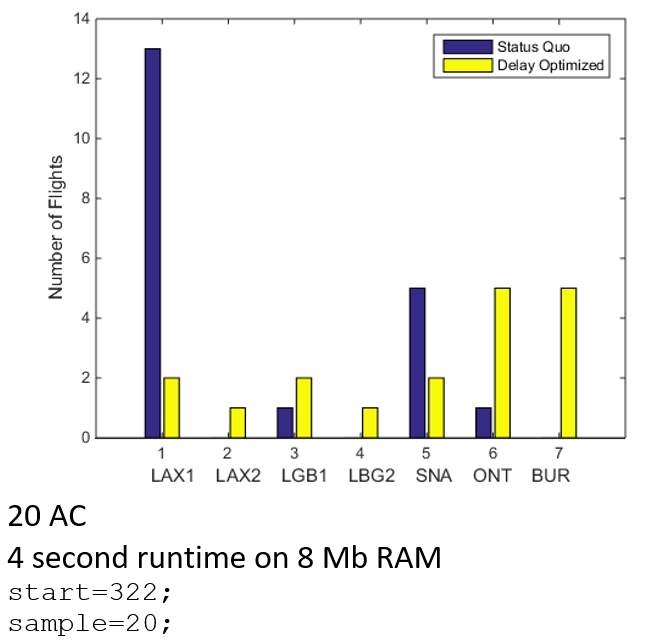
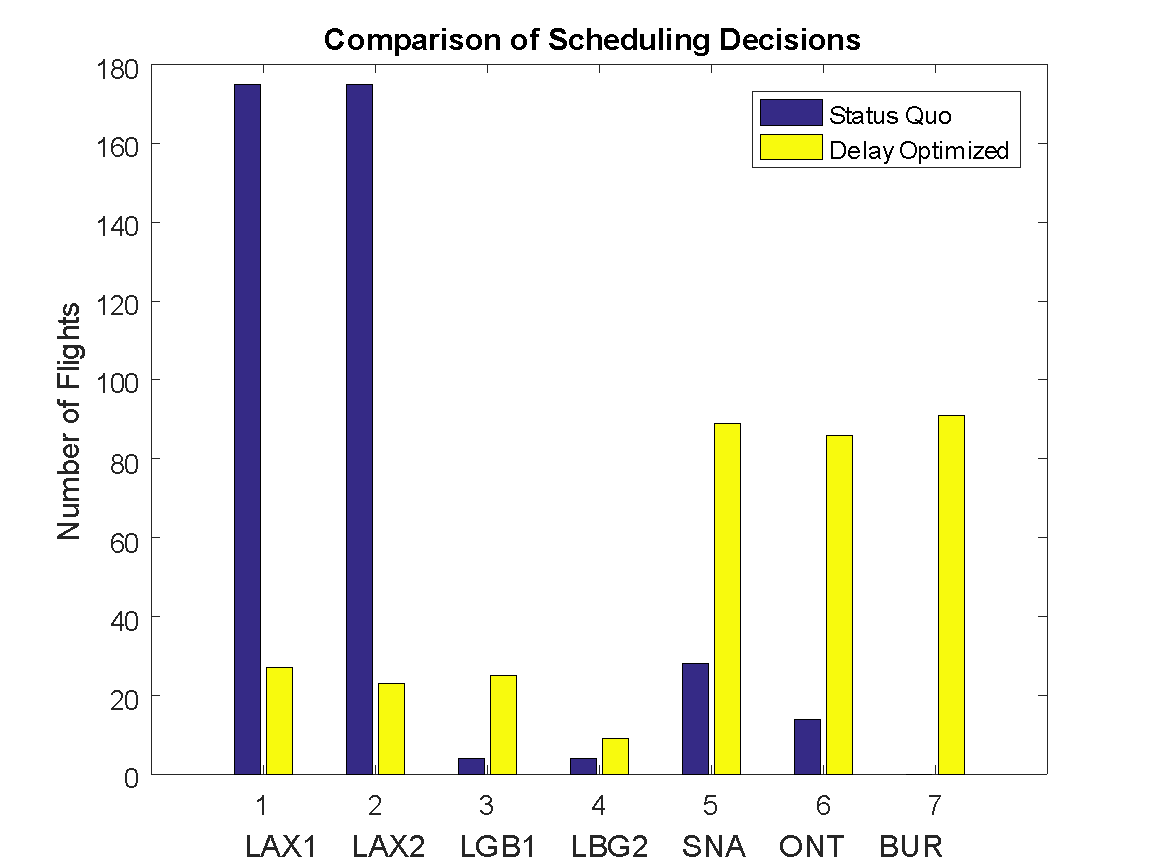
Despite the ability to choose average values for some factors, the permissible scheduling window time proved particularly difficult to determine. This time is defined as the time between the first earliest possible arrival time and the last earliest possible arrival time plus an acceptable delay of X minutes. To determine X, we created another program (LongTermPlot.m) to iterate the scheduler with different values of x as well as different numbers of available runways to schedule on and plot the results. As the following figure shows, a larger number of runways available to schedule over (Active Airports) greatly reduces delays. Delay window (Last Delay Limit (X)) is slightly more complicated in that





**Results**

When operating under ideal conditions, the program can feasibly analyze an interval spanning approximately 45-60 minutes. When these intervals are ran successively, a representation of a peak hour schedule can be developed and a cumulative delay can be produced. This schedule is then represented by a bar chart comparing the original routing of the analyzed aircraft (Status Quo) to the optimized schedule.  
 The results show that current traffic is primairly focused at LAX, but it also shows that an optimized schedule should distribute flights more evenly as hypothesized. This however does not account for their differences in capacity, only in expected delays and geographic separation. Because of this, there is a clear indication that the program is biased towards airports on the periphery of the Los Angeles Basin because they minimize air time and land the aircraft as fast as possible. While this option does avoid air traffic, it does not reflect the fact that landing further from the destination of most passengers will almost always increase their overall carbon footprint by forcing them to use more ground transportation, which is less efficient that air transportation. Capacity constraints and a different system for weighting the priority of airports could be implemented to solve this given additional time.  
 Every iteration generates a plot of runway decisions and a complete schedule vector. Originally sample sizes were too small to produce meaningful results. Most results would be either infeasible or have zero delay. Proper selection of a scheduling window, which was accomplished via LongTermPlot.m, solves this problem.   
 Our program generated a large number of theoretical schedules, all with similar trends. To summarize, two representative examples are shown below. On the left is the largest sample size attempted by our program. On the right is an earlier attempt from a much smaller sample size of aircraft. Their similarity shows how the schedules generated in smaller sample sizes can accurately be extrapolated to larger ones, allowing us to make assumptions about the ~1200 arrivals that occur throughout the day by looking at smaller subsets.



* 200 flights
* 5.5 minutes of delay total
* 7 minute excess scheduling window
* 5 iterations of 40 A/C, starting at 9:00,   
  the typical beginning of peak hour.

|  |  |
| --- | --- |
| Final Results of largest sample size executed | Early results of small sample size |

**Issues and Improvements**

Future improvements upon this program should be in two areas. One is reactivation of certain constraints that were deemed to computationally intensive or too difficult to enter data for. An example of a data entry problem is the issue of categorizing every possible airport of origin and translating its IATA airport code to its approach direction. The other major area is the reduction of time complexity. As our work has shown, accurate representation of real conditions and time complexity must be carefully balanced to produce any useable results.  
 Another possible area of improvement is the inclusion of heuristic search, wherein the program will only assess the nodes that minimize the sum of the best solution yet found and the probable delay of changing to a given runway. This way the function will have to search fewer nodes if, for example, it would only assess the function where a flight is redirected to an uncongested airport rather than checking it at all airports.  
 From the perspective of producing a feasible regulatory strategy, additional data would be needed. As the presentation judging panel pointed out, airlines could greatly increase their efficiency if they consolidated to a single airport, since each airport they have a presence at would require its own minimum amount of staff to operate. Differentiating between international and domestic flights would also be a valuable improvement since it would allow large, hubbing international flights to be directed primarily to LAX while smaller flights could be directed elsewhere due to the high proportions of passengers with Los Angeles as their final destination. Furthermore, comprehensive passenger data on the percent of passengers expecting to transfer would also be needed since spreading them all out would make hubbing inherently impossible in some cases. Such passenger destination data could solve the issue by only allowing the program to change routes with few expected transferring passengers, since they would be prime candidates with few impacts on the traveler experience. Additionally, providing people movers between airports could further change the scenario by allowing passengers to land at one airport and easily transfer at another. However, this would also be an exceedingly large infrastructure project and would require further study.

**Dynamic Programming**

The other approach our team attempted was the creation of a dynamic program scheduler. This however, was met with a long string of technical difficulties in the code due to the much more complicated nature of dynamic programs, and thus the MILP was selected as the most effective option for our project and the best vehicle for further experimentation and improvement. We do however recommend further research into the use of dynamic programming in scheduling, since with more time and resources, it promises to be a more time efficient algorithm that is guaranteed to find a solution without getting bogged down. It is also a prime candidate for the addition of heuristic methods to reduce its time complexity further, making it likely the most efficient option our team is aware of for this problem.

**Computational Conclusions**

As an air traffic software design project, our primary results are a suggested methodology for flight scheduling and the output of our experimental program. The results of our program, though limited, yielded some indication of potential delay reductions. It shows that an MILP optimized schedule can reduce delays from 2.35 minutes (average air delay across the entire ASPM data set sampled) to 0.89 minutes over all final cases analyzed. Experimentation across different scheduling windows and runway options graphically showed that having more runways to schedule over drastically reduces delays and that 7 active runways does in fact have the potential of reducing delays to zero. This also depends on how long a timeframe these aircraft may be scheduled in. Furthermore, the runway decision mix of our experimental program indicates that aircraft should in fact be spread out across all airports but should give priority to smaller periphery airports (SNA, BUR, and ONT) to minimize interaction with LA airspace to reduce possible conflicts with other approaches.   
 For any future research or development of scheduling software, we recommend either using either a MILP run iteratively over several smaller sample sizes or a dynamic program for execution on a larger sample size of aircraft, especially when paired with a heuristic search aspect. Both approaches have their merits, but the majority of our efforts went into investigating the MILP option. Thus, we recommend the following guidelines and factors to account for in future development:

1. Divide aircraft into smaller groups to reduce time complexity and sum the cumulative delays and runway decisions over each successive group.
2. Establish optimality criteria that produce consistent solutions without causing excessively long runtime.
3. Providing a realistic fleet mix (in terms of weight, point of origin, and separation requirements) will improve optimization speed by making some options clearly better than others, leading to faster and more accurate convergence to a solution.
4. Any arbitrary constraints (i.e. constraints not determined by geography, physical limitations, or the FAA) that are up to the discretion of the scheduler should be determined experimentally to find a value that is optimal and always feasible.

**Works Cited**

[1] Alexandre M. Bayen, Claire J. Tomlin, et al. “MILP Formulation and Polynomial Time Algorithm for an Aircraft Scheduling Problem.” *IEEE Conference on Decision and Control* n. pag. Print.

[2] Alexandre M. Bayen, Todd Callantine, et al. “Optimal Arrival Traffic Spacing via Dynamic Programming.” *AIAA Conference on Guidance, Navigation, and Control* n. pag. Print.

[3] Alexandre M. Bayen, and Claire J. Tomlin. “Real-Time Discrete Control Law Synthesis for Hybrid Systems Using MILP: Application to Congested Airspace.” n. pag. Print.

[4] “Federal Aviation Administration.” N.p., n.d. Web. 23 Dec. 2016.

[5] “IATA - Worldwide Airport Slots.” N.p., n.d. Web. 23 Dec. 2016.

**Links**

[1] <http://bayen.eecs.berkeley.edu/sites/default/files/conferences/MILP_formulation.pdf>

[2] <http://bayen.eecs.berkeley.edu/sites/default/files/conferences/gnc04.pdf>

[3] http://bayen.eecs.berkeley.edu/sites/default/files/conferences/real-time\_discrete\_control\_law\_synthesis.pdf

[4] <https://aspm.faa.gov/>

[5] <http://www.iata.org/policy/infrastructure/slots/Pages/index.aspx>

**Appendix**

1. MILP Scheduler (LAdelay.m)
2. Bivariate constraint plotting script/scheduler iteration mechanism (LongTermPlot)
3. Dynamic Program Scheduling outline (DynLAdelay.m)

|  |
| --- |
| LAdelay.m (A) |
| %A Forced Market Model for Delay Minimization in the LA Basin(J^4 + B)  %Title: LAdelay    %Function: Minimize delays over a given time span by manipulating  %destination of each aircraft (bin), as well as # of holds (int)  % and construction of additional capacity (bin).  %Then calculate expected cost of redirection for each airport and airline  %(in terms of lost passengers/landing fees and change in fuel burn,  %possibly including social cost of carbon) and determine an effective level  %of incentives for implementing the optimum. (or assume airports share  %profits for additional simplification i.e. LAWA annexes everything)    %Load individual flight data for given time span (peak demand hour)  ARR=importdata('ARR.csv'); %(flights x data types)  ARRn=ARR.data;  ARRt=ARR.textdata;    %%DEP=importdata('DEP.csv');  %DEPn=DEP.data;  %DEPt=ARR.textdata;    start=322+sample\*(NG-1);    N=length(ARRn(start:start+sample-1,1));  R=7;  Rcap=[3; %LAX1 %----additional runway analysis required----  3; %LAX2  2; %LGB1  2; %LGB2  2; %SNA  2; %ONT  2]; %BUR  size = ones (N,1); %create A/C size matrix  %for i=1:N %find the size of the aircraft based on tail number  % if or(strcmp(ARRt(i,3),[TN, TN, TN])); %must be cell array  % Size(i) = 1; %heavy  % elseif or(strcmp(ARRt(i,3),[TN, TN, TN]));  % Size(i) = 2; %B757  % elseif or(strcmp(ARRt(i,3),[TN, TN, TN]));  % Size(i) = 3; %large  % else  % Size(i) = 4; %small  % end  %end    %create matrix of FAA separation regulations based on A/C size  FAAsep=(1/3600).\*[90 120 120 120; %rows = H->S leading in minutes  90 90 90 120; %cols = H->S following  60 60 60 60;  45 45 45 45];  T=zeros(N,N);%create minimum headway Matrix T  for i=1:N %create matrix of lead/follow distance relationships between each A/C  for j=1:N  T(i,j)=FAAsep(size(i),size(j));  end  end    %calculate expected air time assuming delays at point of origin and enroute  %time are known. Total airtime represents the expected time of arrival with  %acceptable delays. This program minimizes unacceptable delays in LA airpspace which is  %anything beyond what the airline scheduled for  te= ARRn(start:start+sample-1,18); %ARRn(start:start+sample-1,2)+ARRn(start:start+sample-1,5)./60+ARRn(start:start+sample-1,17)./60; %te=t\_gateOutActual + t\_TaxiOutActual + ATe    %ARRerr=te+(1/60)\*floor(3\*randn(length(te),1)); %Arr = data(:,1) + rand..... random error    M=1e3; % Big M    ncons=(2\*N+N\*(N-1)+R); %number of constraints    c=zeros(3\*N+N\*R+N\*(N-1)/2,1); %form cost vector  nvars=length(c); % number of variables  c(1:N)=ones(1,N); % minimize sum of delayed arrival times    Tdest= (1/60).\*[5 5 10 10 15 5 0;  10 10 10 10 10 0 10;  10 10 5 5 0 10 15;  0 0 0 0 0 10 0]; %time to each destination in minutes    ActDest=zeros(N,1); %initialize actual destination  Origin = randi(4,N,1);    ARRt(start:start+sample-1,:)  ARRn(start:start+sample-1,:)  for i = 1:N  ActDest(i)= 1\*strcmp(ARRt(i+1+start,6),'LAX')+3\*strcmp(ARRt(i+1+start,6),'LGB')+ 5\*strcmp(ARRt(i+1+start,6),'SNA')+ 6\*strcmp(ARRt(i+1+start,6),'ONT')+7\*strcmp(ARRt(i+1+start,6),'BUR'); %determine original destination  % if strcmp(ARRt(i+1+start,5),'orig'); %cell array of eastern origins %determine original direction of origin  % Origin(i)=1; %North  %elseif strcmp(ARRt(i+1+start,5),'orig'); %cell array  % Origin(i)=2; %East  %elseif strcmp(ARRt(i+1+start,5),'orig'); %cell array  % Origin(i)=3; %South  %else  % Origin(i)=1; %West %change later from 1 to 4  %end  end  %% Additional Constraints beyond A and Aeq  holdlim=0; %set max number of holds  ERdelaylim=0; %set max amount of en route delay  DelayLim=LM/60;% 10/60; %delay limit for any aircraft in hours    lb = zeros(length(c),1); %lower bound on all variables  lb(1:N)= ARRn(start,18);  ub = [(ARRn(start+sample-1,18)+DelayLim)\*ones(N,1); ones(N\*R+N\*(N-1)/2,1);holdlim(ones(N,1));ERdelaylim(ones(N,1))]; %upper bound on all variables  offLimits = (7-OL):7;%Set off limit runways  for i = offLimits  ub(N+i:R:N+R\*(N-1)+i) = 0;%impose upper limit constraints on off limits runways  end  intcon = [N+1:length(c)]'; %specify which variables are integer  H=1.5/60; %holding pattern length in hours    binCo=[-1 1]'; %most common block of coefficients in A matrix    %% Clear Source and Initialize Arrays  clear ARR\*    Aeq=zeros(2\*N+R,length(c)); %initialize Aeq matrix  A=zeros(2\*N+R\*N\*(N-1),length(c)); %initialize A matrix    beq=[zeros(length(te),1);ones(N,1);zeros(R,1)]; %initialize beq  b=[te+DelayLim;-te;zeros(R\*N\*(N-1),1)]; %initialize b    %% Assemble time constraints  Atcons=zeros(N,length(c));  Atcons(1:N,1:N)= -diag(ones(N,1)); %assemble t components of tcons  %Aeq(1:N,length(c)-2\*N+1:length(c)-N)=-H\*diag(ones(N,1)); %assemble H components of tcons  %Aeq(1:N,length(c)-N+1:length(c))=-diag(ones(N,1)); %assemble D components of tcons  A(1:N,1:N)=diag(ones(N,1)); %assemble delay limits    for i= 1:N %find additional travel time for change of destination  Atcons(i,N+R\*(i-1)+1:N+R\*i)= (Tdest(Origin(i),:)-Tdest(ActDest(i))); %assemble AT portions of tcons for NSEW  Aeq(N+i,N+R\*(i-1)+1:N+R\*i)=ones(1,7); %assemble single destination constraints for each A/C  end    %A(N+1:2\*N,1:N)=-diag(ones(N,1)); %assemble minimum arrival times  A(N+1:2\*N,:)=Atcons(1:N,:); %assemble minimum arrival times    %% runway capability constraints  %for r=1:R  %  %end    %% Order Constraint Assembly    MRow=1; %start with first pair of Big M constraint rows (Mcons)  for i=1:N %assemble t components of all time interval constraints (tcons)    for k=1:N  if i>k  Tco=[T(i,k);T(k,i)];  A((2\*N+2\*R\*(MRow-1)+1:2\*N+2\*R\*MRow),i)=repmat(binCo,[R,1]); %assemble t components of Mcons  A((2\*N+2\*R\*(MRow-1)+1:2\*N+2\*R\*MRow),k)=-repmat(binCo,[R,1]); %assemble t components of Mcons  A((2\*N+2\*R\*(MRow-1)+1:2\*N+2\*R\*MRow),N+R\*(i-1)+1:N+R\*i)=blkdiag(Tco,Tco,Tco,Tco,Tco,Tco,Tco);  A((2\*N+2\*R\*(MRow-1)+1:2\*N+2\*R\*MRow),N+R\*(k-1)+1:N+R\*k)=blkdiag(Tco,Tco,Tco,Tco,Tco,Tco,Tco);  b(2\*N+2\*R\*(MRow-1)+1:2\*N+2\*R\*MRow)=repmat([T(i,k);M+T(k,i)],[R,1]); %assemble order constraints in beq  A(2\*N+(2\*R\*(MRow-1)+1:2\*R\*MRow),N+N\*R+MRow)=M\*repmat(binCo,[R,1]);  MRow=MRow+1; %iterate to next constraint set  end  end  end    %for i=1:R\*N\*(N-1)/2 %Assemble right hand side of Mcons    %end        %% Solve MILP    Sched=intlinprog(c,intcon,A,b,Aeq,beq,lb,ub); %solve MILP  if isempty(Sched)  TotalDelay=inf;  else  Diff=Sched(1:N)-te;  TotalDelay=sum(Diff(Diff>0));  timespan=ub(1)-lb(1);  AveTimespan=(timespan\*(NG-1)+AveTimespan)/NG;  end      %% Plot Results  %figure;  %StatQuo=zeros(1,R);  %NewDest=zeros(1,R);  %for i=1:7  % StatQuo(i)=sum(ActDest==i);  % NewDest(i)=sum(Sched(N+i:R:N+N\*(R-1)+i));  %end  %bar(1:7,[StatQuo',NewDest'])  %ylabel('Number of Flights')  %xlabel('LAX1 LAX2 LGB1 LBG2 SNA ONT BUR')  %legend('Status Quo', 'Delay Optimized')    %% Construct AP assignment on long term basis    for i=1:7  StatQuo(7-OL,LM-2,i)=sum(ActDest==i)+StatQuo(7-OL,LM-2,i);  if not(isempty(Sched))  NewDest(7-OL,LM-2,i)=sum(Sched(N+i:R:N+N\*(R-1)+i))+NewDest(7-OL,LM-2,i);  end  end |

|  |
| --- |
| LongTermPlot.m (B) |
| %% Long term Plot  clear;  LimitMins=3:10;  Delay3Var=zeros(7,length(LimitMins));  S=size(Delay3Var); %total number of scenarios  NumGroups = 5;  AveTimespan=0;  sample=6;  StatQuo=zeros(7,length(LimitMins),7);  NewDest=zeros(7,length(LimitMins),7);    figure;  hold off;  xlim([0 7]);  ylim([3 10]);  title(sprintf('Delay for 25 A/C over %d samples w.r.t runway options and delay limits',NumGroups));  zlabel('Delay (minutes)')  ylabel('Last Delay Limit (minutes)')  xlabel('LAX1 LAX2 LGB1 LBG2 SNA ONT BUR')  hold on;  for NG = 1:NumGroups  for OL = 0:6  for LM = LimitMins  LAdelayV2  Delay3Var(7-OL,LM-2)= TotalDelay\*60 + Delay3Var(7-OL,LM-2);  plot3(7-OL,LM,Delay3Var(7-OL,LM-2),'ro')  end  end  end  %% Surf Plot    figure;  ylim([0 7]);  xlim([3 10]);  surf(3:10,1:7,Delay3Var)  title(sprintf('Delay for %d A/C over %d samples w.r.t runway options and delay limits',sample,NumGroups));  zlabel('Delay (minutes)')  xlabel('Last Delay Limit (minutes)')  ylabel('Active Airports')    figure;  ylim([0 7]);  xlim([3 10]);  surf(3:10,1:7,Delay3Var./(NumGroups\*sample))  title(sprintf('Delay per A/C for %d A/C over %d samples w.r.t runway options and delay limits',sample,NumGroups));  zlabel('Delay per A/C (minutes)')  xlabel('Last Delay Limit (minutes)')  ylabel('Active Airports')    %% Plot Results Sorted by Airport for minimum delay scenario  [Y, row]= min(Delay3Var);  [YY, col]= min(Y);    figure;    SQvec=zeros(7,1);  NDvec=zeros(7,1);    for i=1:7  SQvec(i)=StatQuo(row(col),col,i);  NDvec(i)=NewDest(row(col),col,i);  end    bar(1:7,[SQvec,NDvec])  ylabel('Number of Flights')  xlabel('LAX1 LAX2 LGB1 LBG2 SNA ONT BUR')  legend('Status Quo', 'Delay Optimized')  legend('Status Quo', 'Delay Optimized')  title('Flights to each runway over all time intervals')  AveTimespan |

|  |
| --- |
| DynLAdelay.m (general layout of the dynamic solver portion) (C ) |
| . . . .  %% Solve Dynamic Program  for k = Tl:-1:Tf %or T:-1:1  for r=1:R  PosInd=find(te\_ir<k,te\_ir);  [tiLast,idL]=min(te(find(Tstar(r,k+1,:))+Tstar(find(Tstar(r,k+1,:))))); %determine last arrival time for runway  for i=PosInd % this should run out if all AC have landed.  if te\_ir(i,r) < (tiLast-FAAsep) %only optimize if min operation time is within time horizon  %if OpType=1; %arrival vs departure conditions    step\_diff = [te\_ir:(tiLast-FAAsep)]-te(i);  TotalDelay=sum(step\_diff(step\_diff>0)); %cost per time step    %determine index among all R and A and indnxt from Tstar  ind = sub2ind([R,A],r,i);  ind\_nxt = idN(1);    %principle of optimality  [V(ind,k,i), idx] = min([V(ind,k+1,i); c + V(idL)]); %??????? ind\_nxt,k+1,i    % else    % end    end  end  end  end |