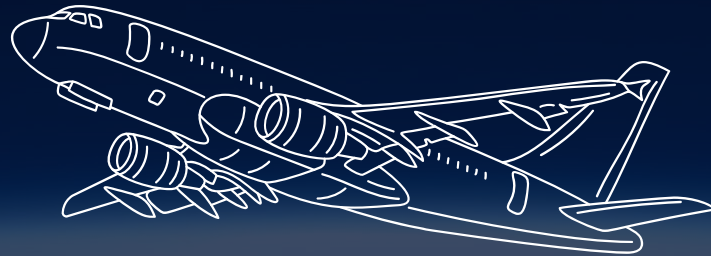


# OptiFlight

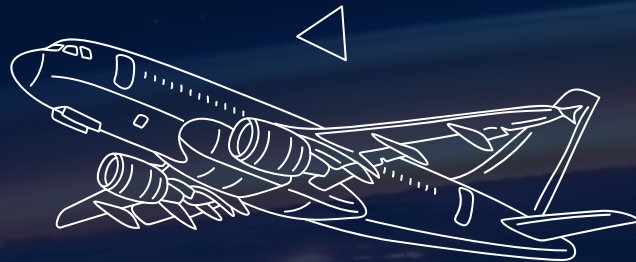


Real-Time Flight Recovery Optimization for Post-Disruption Scheduling  
*Team 72 - The Mahwah Masters*

01

# Introduction

Who are we?



# Background About Us...

Vidyut Rajagopal  
Aadit Mital  
Vikas Shah  
Rohan Machhi  
Michael Matwiejczuk



# The Problem

During significant weather events, flight operations teams in Louisville, KY must manually re-plan every single affected flight while enforcing complex crew, aircraft, and regulatory constraints!



# What Makes This Problem Complex



## Crew

duty cycles,  
availability, rest windows



## Aircraft

availability, maintenance,  
type limitations



## Cargo Priority

time-sensitive shipments  
(Next Day Air, medical)



## Airport

gate compatibility, fuel  
access



## Certification

crew-aircraft pairing rules,  
FAA compliance, type  
ratings



## Real-Time Disruptions

dynamic weather, crew  
no-shows, mechanical  
failures

# Why is this a problem?



## Time Consuming

Flight operations teams spend thousands of man hours per year manually navigating this process



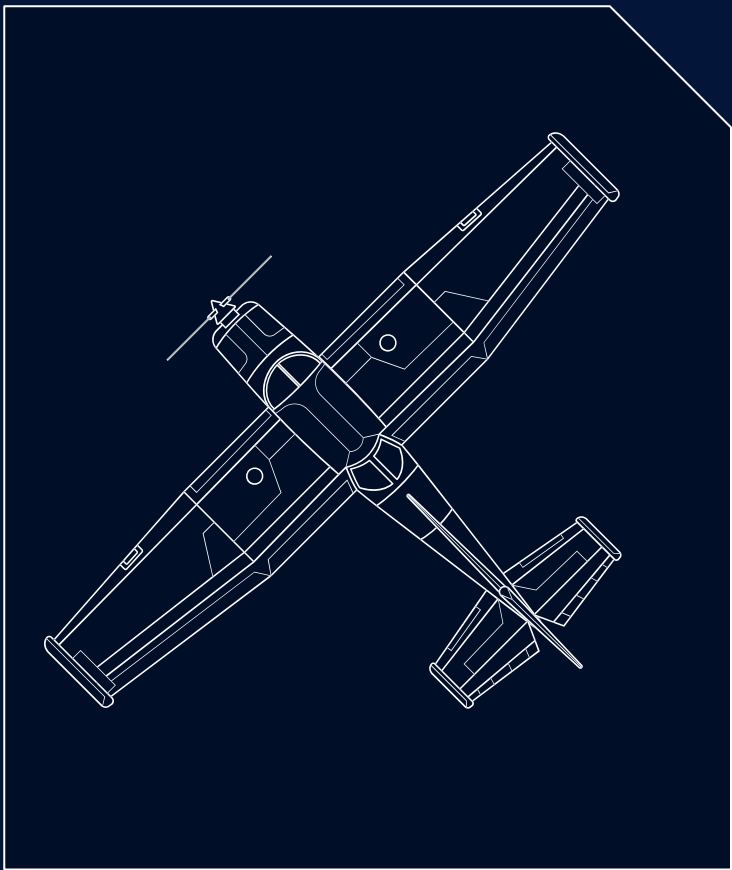
## Inefficient

Manual human solutions can never create highly optimal solutions



## Never ending

On-ground situations are constantly changing and require ongoing resolution



## Our Solution:

An optimization-driven solution that automates outbound flight plans in response to real-time weather disruptions.





# Brief Solution Overview



Real time data  
ingestion

Sources millions of  
live data points  
through real time  
API

Feeding the  
model

Passes processed  
data points into  
the optimization  
engine

Optimization

Uses multi-thread  
computing and  
node analysis to  
determine the  
optimal output

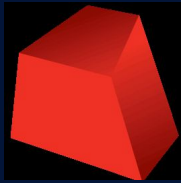
Continuous  
Feedback

Allows the user to  
easily edit/refine  
constraints and  
generate new  
optimals





# Driving Technology



**GUROBI**  
OPTIMIZATION



Gurobi is a state-of-the-art  
**mathematical optimization solver** used in  
logistics, finance, and energy to find the  
best solution from millions of  
possibilities - in real time.

+

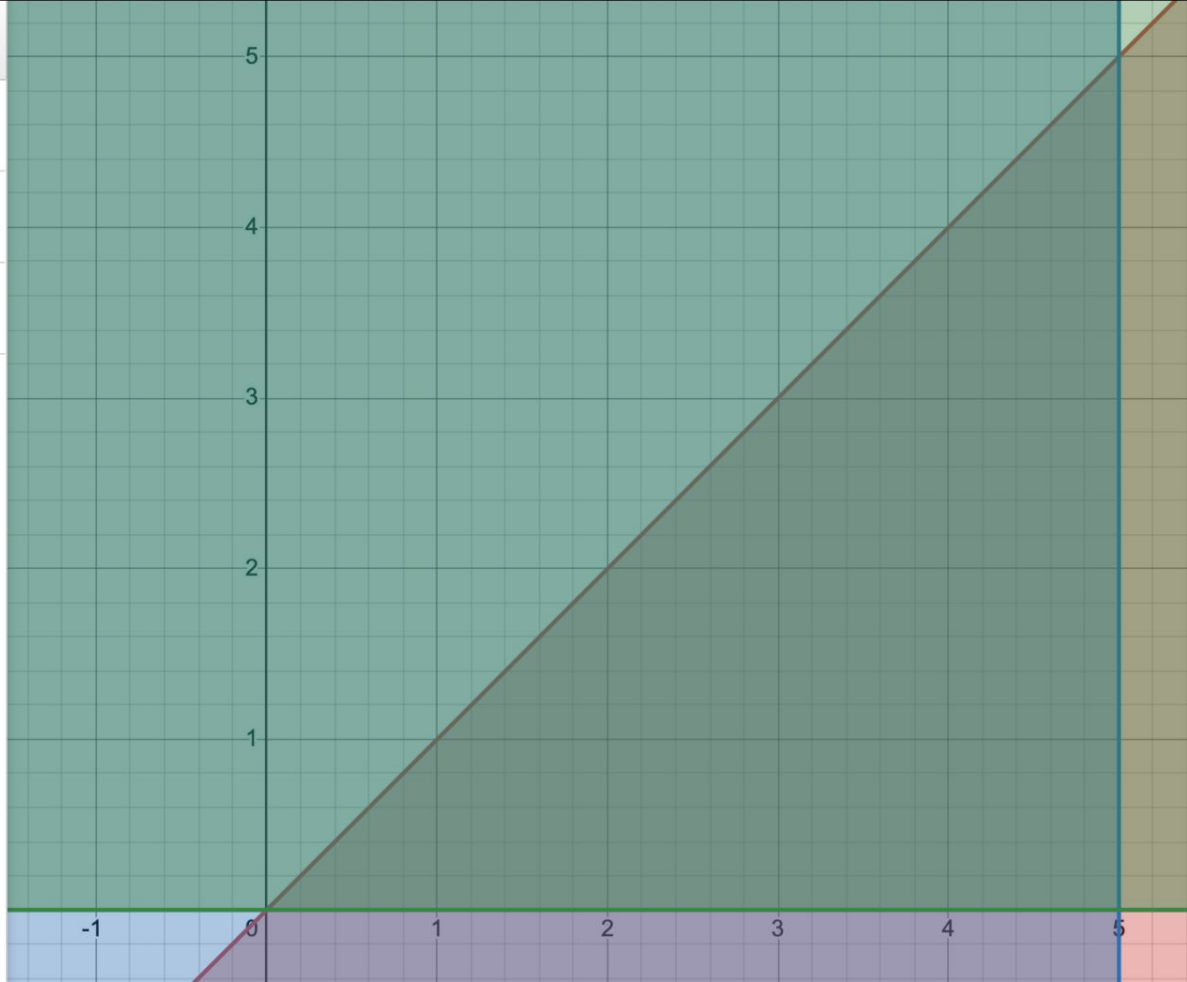
1  $y \leq x$

2  $x \leq 5$

3  $y \geq 0$

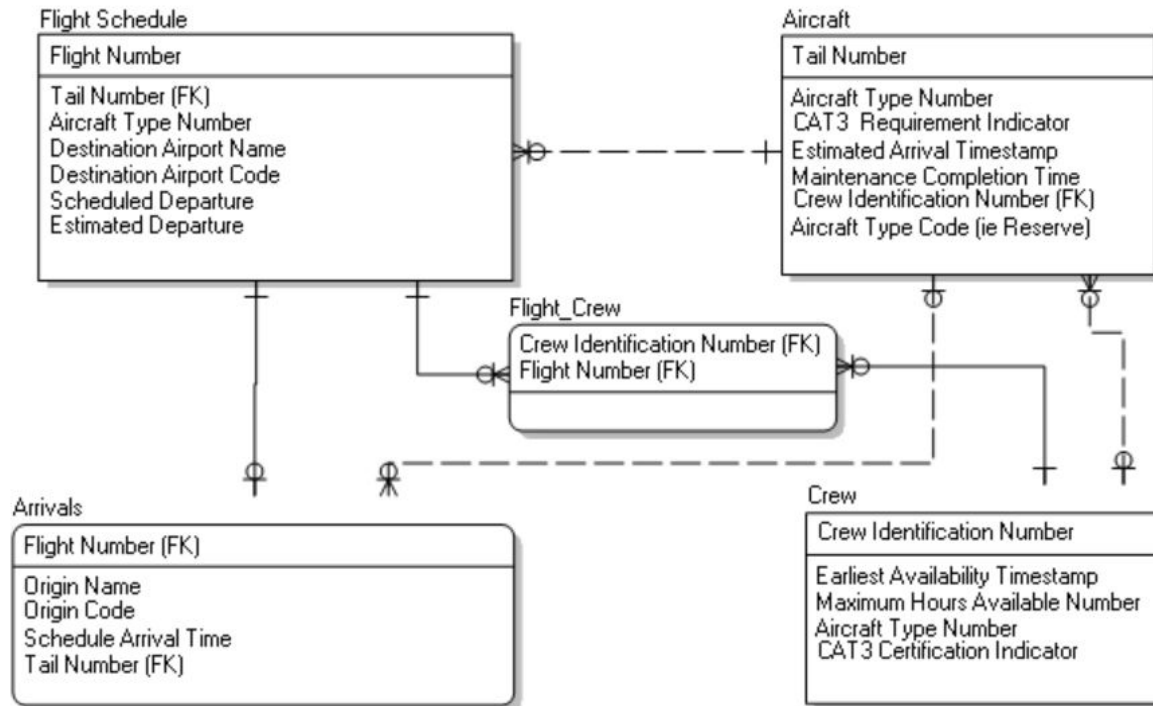
4

Objective: Maximise Y



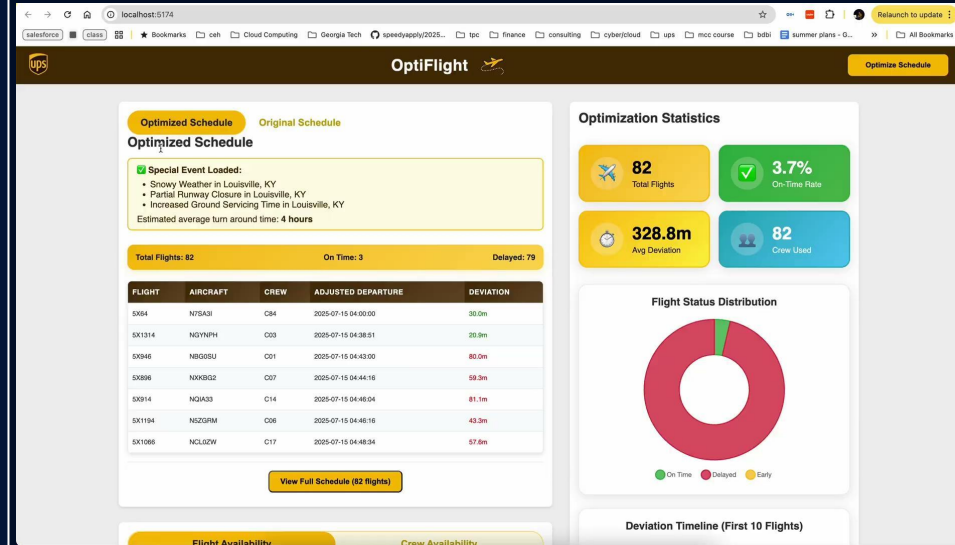
# Data Overview

## Opti-Flight





# Solution Demo



THE-MAHWAH-MASTERS

api

arrivals\_api.py

departures\_api.py

flight\_scheduler

\_\_pycache\_\_

data

chatbot\_responses.txt

csv\_chatbot.py

generate\_aircraft.py

generate\_crew.py

generate\_schedule\_report.py

opt.py

optimization\_stats.csv

optimized\_schedule.csv

serve\_schedule.py

unavailable\_crews.txt

my-react-app

public

src

assets

components

pages

utils

App.css

App.jsx

Index.css

main.jsx

.gitignore

eslint.config.js

index.html

package-lock.json

package.json

README\_FLIGHT\_DETAILS.md

README.md

vite.config.js

api-server.js

package.json

README.md

requirements.txt

run\_optimization\_api.py

run\_optimization\_simple.sh

flight\_scheduler > opt.py > ...

1 import pandas as pd

2 import gurobipy as gp

3 from gurobipy import GRB

4

5 import os

6

7 # 1. Load raw CSVs

8 ac = pd.read\_csv('data/aircraft.csv')

9 fl = pd.read\_csv('data/schedule.csv')

10 cr = pd.read\_csv('data/crew.csv')

11

12 # 1a. Preserve original departure time

13 fl['OriginalDepartureTime'] = fl['ScheduledDeparture']

14 cr['OriginalEarliestAvail'] = cr['EarliestAvail']

15

16 # 2. Parse all date/time columns using ISO8601

17 for col in ['EstimatedArrival', 'MaintenanceDoneTime']:

18 ac[col] = pd.to\_datetime(ac[col], format='ISO8601')

19 for col in ['ScheduledDeparture', 'EstimatedDeparture']:

20 fl[col] = pd.to\_datetime(fl[col], format='ISO8601')

21 fl['OriginalDepartureTime'] = pd.to\_datetime(fl['OriginalDepartureTime'])

22 cr['EarliestAvail'] = pd.to\_datetime(cr['EarliestAvail'], format='ISO8601')

23 cr['EarliestAvail'] = cr['EarliestAvail'] + pd.Timedelta(hours=4)

24

25 # 3. Compute each aircraft's availability time

26 ac['AvailableAt'] = ac[['EstimatedArrival', 'MaintenanceDoneTime']].max(axis=1) + pd.Timedelta(hours=4)

27

28 # 4. Prepare index lists

29 aircrafts = ac.index.tolist()

30 flights = fl.index.tolist()

31 crews = cr.index.tolist()

32

33 # 5a. Flag 747-only flights

34 fl['Requires747'] = fl['AircraftType'].str.contains('747', case=False, na=False)

35 # 5b. Flag CATIII flights (placeholder until you derive from your data)

36 fl['RequiresCAT3'] = False

37

38 with open('unavailable\_crews.txt') as f:

39 # strips whitespace/newlines, ignores blank lines

40 blocked\_ids = [line.strip() for line in f if line.strip()]

41 #print(blocked\_ids)

42

43 # 6. Build model and decision variables

44 m = gp.Model('aircraft\_and\_crew\_scheduling')

45 x = m.addVars(aircrafts, flights, vtype=GRB.BINARY, name='assign\_ac')

46 valid\_crews = [c for c in crews if cr.at[c, 'CrewId'] not in blocked\_ids]

47 y = m.addVars(valid\_crews, flights, vtype=GRB.BINARY, name='assign\_cr')

48

49

50 # 6b. --- Flexible-departure variables ---

51 # pick a reference timestamp

Repo Overview

Optimize button on React Frontend

Backend Orchestration (Flask Server)

Triggers a shell script to kick off full pipeline

Step-by-Step Data Flow

1. Live API Call

2. Pre-Optimization (cleans + normalizes flight data)

3. Optimization Engine (Gurobi)

4. Post-Processing (parse result to frontend)

# Unlimited Customization



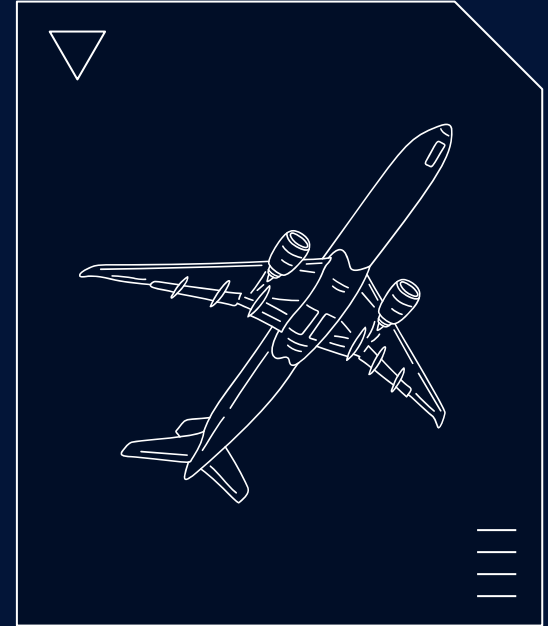
## Objectives

Any number of objectives can be considered by the algorithm



## Priority Adjustment

Users can edit the weightage of each priority in real time





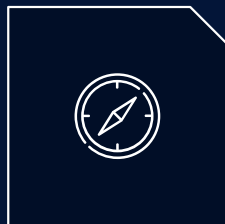
## Crew

Crew unable to make it  
due to icy roads



## Aircraft

Unexpected  
maintenance



## ATC Restrictions

Ground stop until 5 pm

**Optimize Schedule**



# Financial Implication

Before Our Solution	After Our Solution
Unnecessary departure delays	\$10-15M/year in SLA penalties, and reputation value saved
Unnecessary costs for crew/aircraft relocation	Estimated \$1-5M/year in operational savings
Thousands of man hours spent	Cuts time spent by over 90% unlocking labor efficiencies





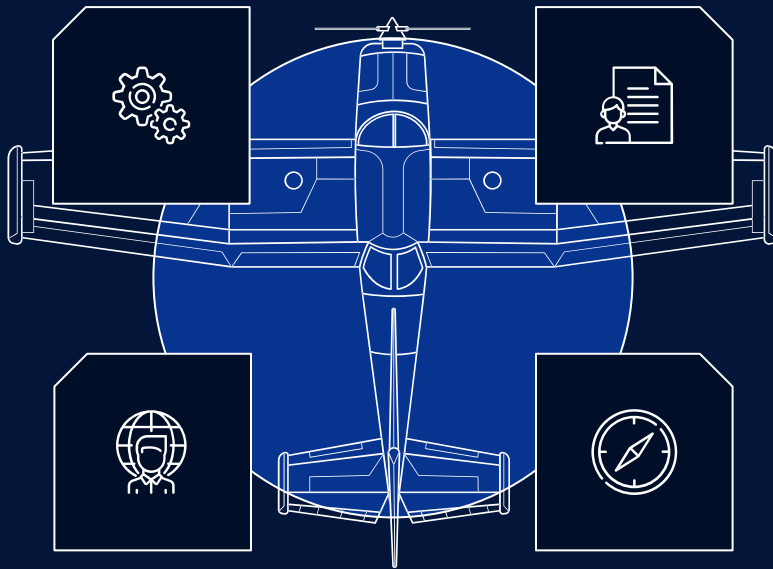
# Future Integrations

Ground Systems

Pipeline to automate  
flight plan  
submissions

Integrate with  
ARDAP/ARR

Critical  
Package  
Visibility Tool





Thank  
you