



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

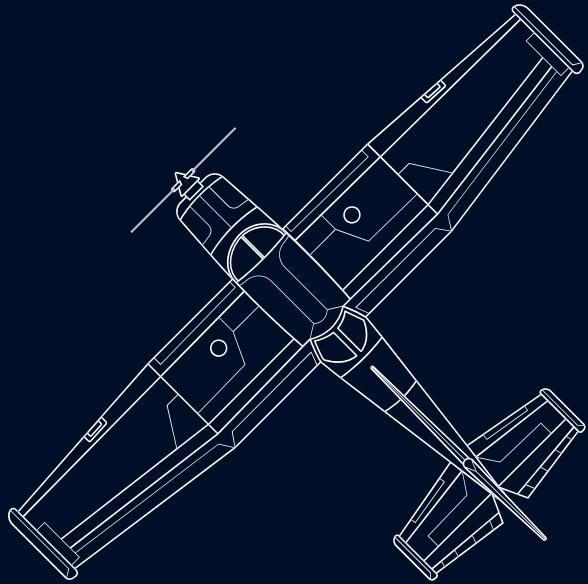
Manual human solutions can never create highly optimal solutions



## 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

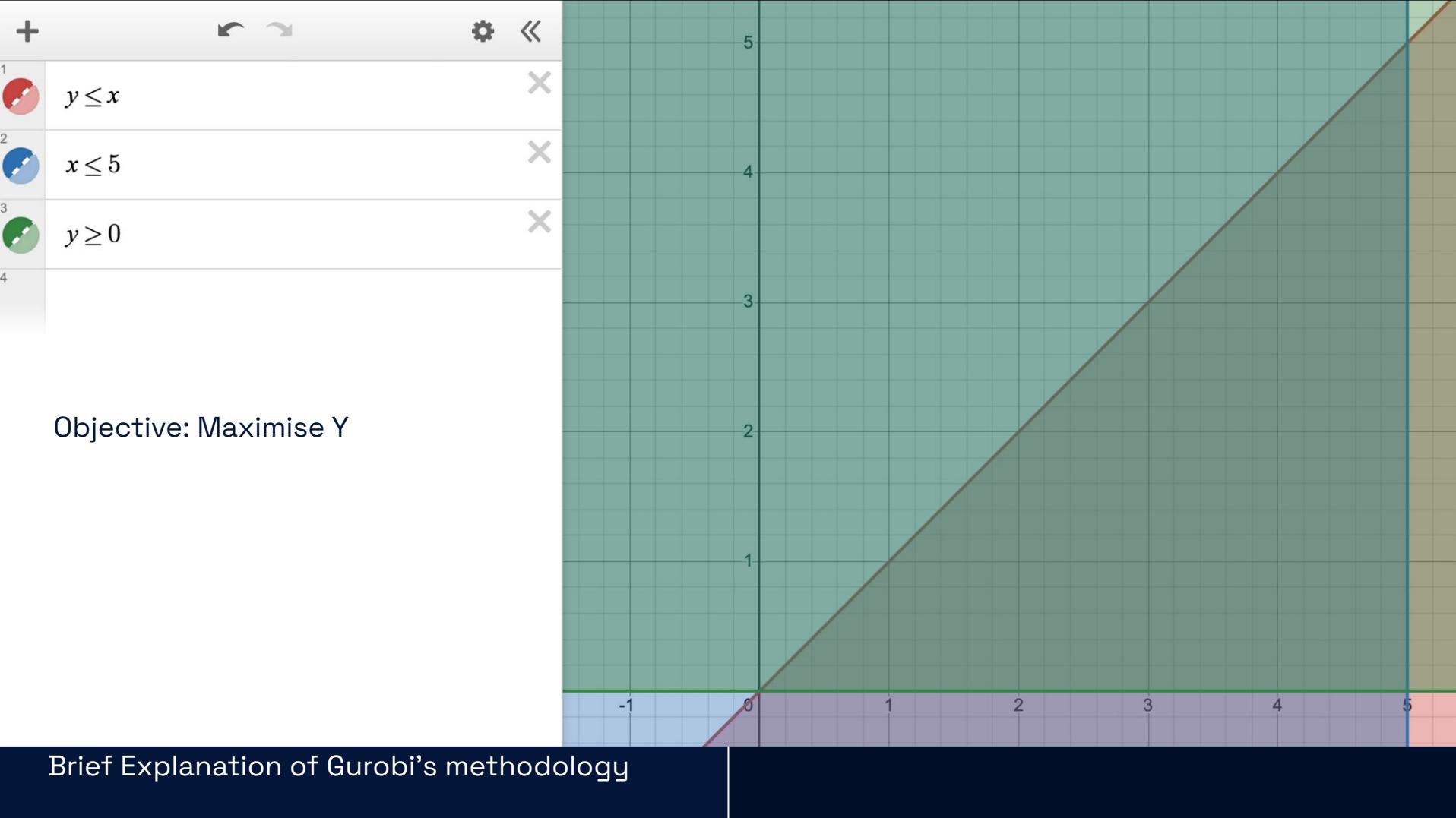
## Continuous Feedback

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

# Driving Technology

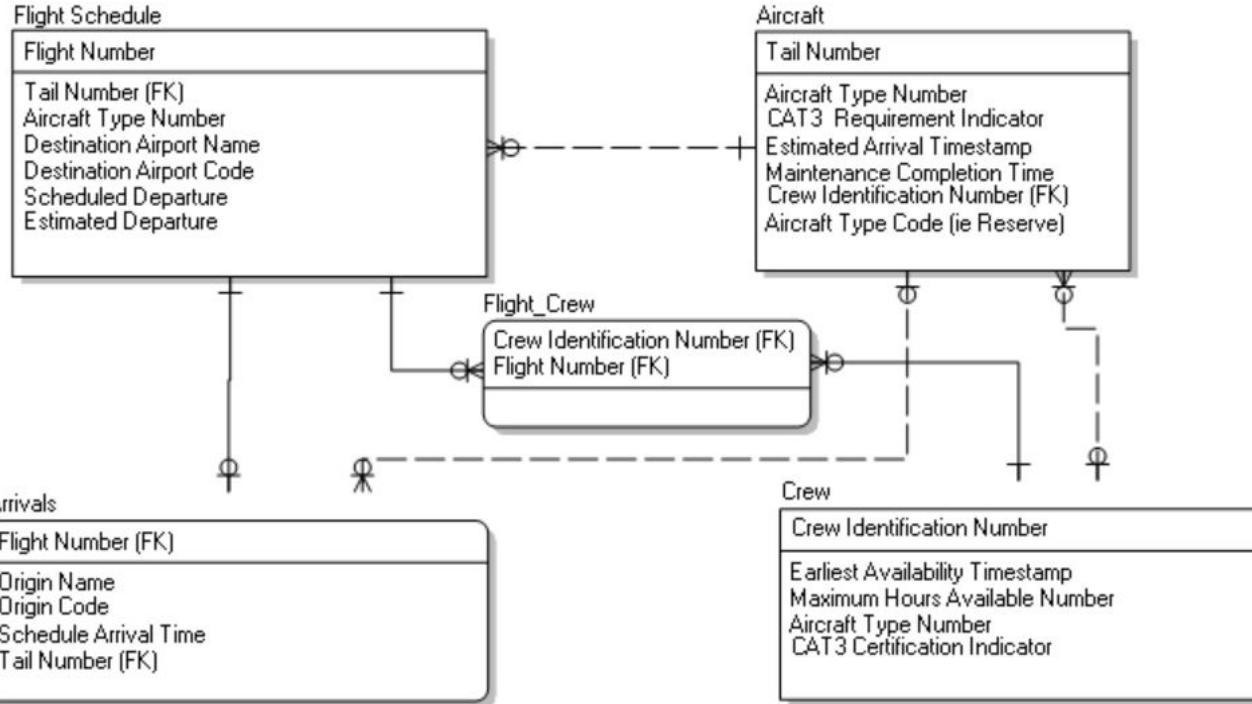


Gurobi is a state-of-the-art **mathematical optimization solver** used in logistics, finance, and energy to find the best solution from millions of possibilities - in real time.



# Data Overview

## Opti-Flight





# Solution Demo

localhost:5174

Relaunch to update

OptiFlight

Optimized Schedule   Original Schedule

Special Event Loader  
• Snow Storm in Louisville, KY  
• Partial Runway Closure in Louisville, KY  
• Increased Ground Servicing Time in Louisville, KY  
Estimated average turn around time: 4 hours

Total Flights: 82   On Time: 3   Delayed: 79

FLIGHT	AIRCRAFT	CREW	ADJUSTED DEPARTURE	DEVIATION
5X64	N7SAJ3	C84	2025-07-15 04:00:00	30.0m
5X1314	NGYNPH	C03	2025-07-15 04:38:51	20.9m
5X946	NBIGO8U	C01	2025-07-15 04:43:00	80.0m
5X096	NXXBG02	C07	2025-07-15 04:44:16	59.3m
5X914	NQUA33	C14	2025-07-15 04:46:04	81.1m
5X1194	NGZGORM	C08	2025-07-15 04:46:11	43.3m
5X1066	NCLGZ2W	C17	2025-07-15 04:48:34	57.6m

[View Full Schedule \(82 flights\)](#)

Optimization Statistics

82 Total Flights	3.7% On-Time Rate
328.8m Avg Deviation	82 Crew Used

Flight Status Distribution

Deviation Timeline (First 10 Flights)

Flight Availability   Crew Availability

# Repo Overview

```
THE-MAWHAW-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
      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
```

- Optimize button on React Frontend

## 🧠 Backend Orchestration (Flask Server)

- Triggers a shell script to kick off full pipeline

## 🔗 Step-by-Step Data Flow

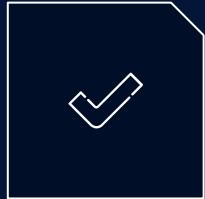
- Live API Call
- Pre-Optimization (cleans + normalizes flight data)
- Optimization Engine (Gurobi)
- Post-Processing (parse result to frontend)

# Unlimited Customization



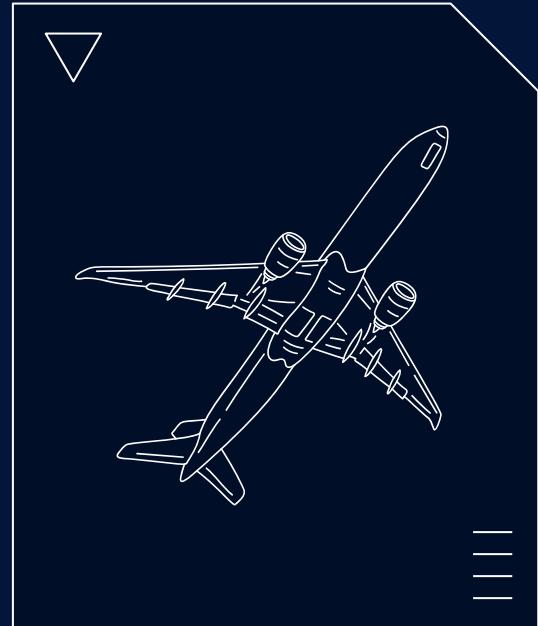
## Objectives

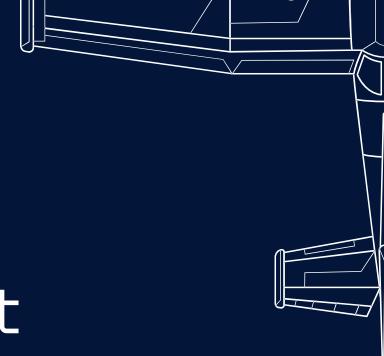
Any number of objectives can be considered by the algorithm



## Priority Adjustment

Users can edit the weightage of each priority in real time





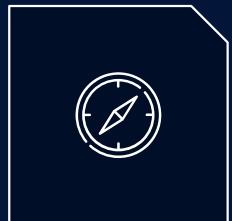
## Crew

Crew unable to make it  
due to icy roads



## Aircraft

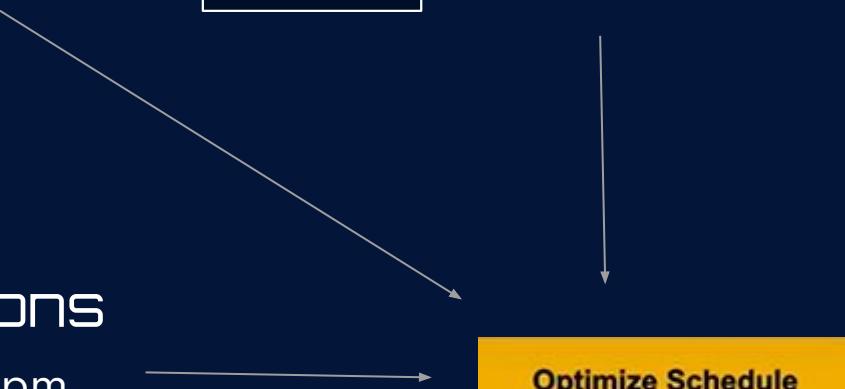
Unexpected  
maintenance



## 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



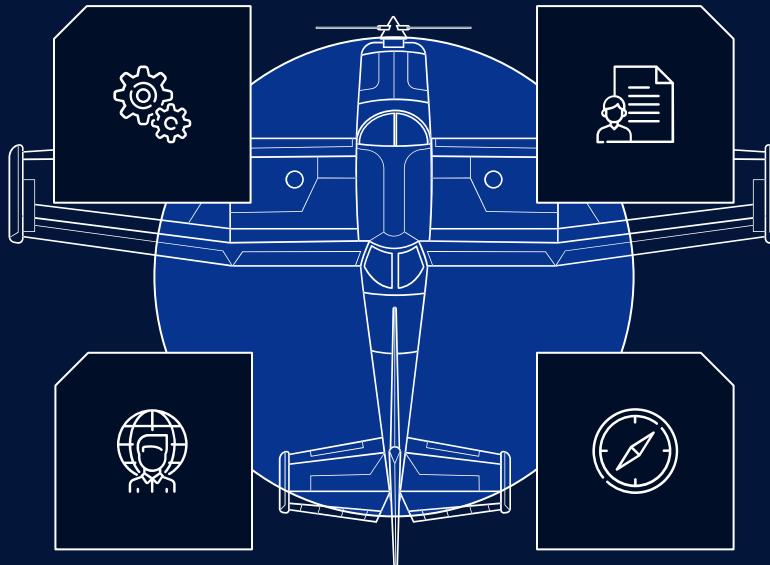
# Future Integrations

Ground Systems

Integrate with  
ARDAP/ARR

Pipeline to automate  
flight plan  
submissions

Critical  
Package  
Visibility Tool





Thank  
you

