Group Members: Kenneth Gutierrez King (gutierrezking.1@buckeyemail.osu.edu), Ayaan Baig (baig.59@buckeyemail.osu.edu), Shan He (he.2212@buckeyemail.osu.edu)

Concert Scheduling Based on Optimal Flights

Airline tickets are often susceptible to price gouging attempts by travel sites. For people trying to travel, the destination and time of ticket purchases can set these travel sites to try and push higher prices than what would be a reasonable price. For example, a standard economy flight to Kyoto from Cleveland, Ohio can cost up to $2,500 if the ticket is purchased on a weekend some time in June. A similar flight may cost much less on a Monday morning, and even less in the month of May. The cost of a ticket can then be further reduced by flying to a different airport, for example a flight from Cleveland Hopkins airport to Logan airport, Chicago to Haneda airport, Tokyo to Itami airport, Osaka (1 hour from Kyoto) is far cheaper than a flight from Cleveland airport to JFK airport, New York to Osaka International Airport (Also 1 hour from Kyoto). As for the scope of our proposal, we would like to minimize the cost of a hypothetical artist trying to buy tickets while performing a world tour.

One can see how difficult it would be to schedule several flights to several cities in a short time frame. This difficulty is often a reality for people in industries that require heavy travel, and minimizing costs for scheduling these trips becomes an even more tedious task. An example of this problem in the real world, and the example we chose for this report, is scheduling concerts for a famous star. For our scenario, we are Kanye's team of world tour planners, but due to the star's recent controversies, he is unable to afford a private plane for us. Our objective became to ensure Mr. West has his world tour in his desired locations while also minimizing the costs of our own plane tickets to each of these locations.

Decision Variables

Two key decision variables. Our first decision variable would be $\mathbf{X_{i,\,c,\,d}}$ which is a binary indicator of which flight is used, where i is the flight number (13,020 different flights in the dataset). Our second decision variable is $\mathbf{Y_{c,d}}$, where a concert in city c is scheduled on date d. Some of the possible constraints that we could see are that there is only one concert per city, The artist can stay at one location for at most 3 days and at least 2 days.

Dataset:

In terms of data, we have created a flight dataset using Generative AI that contains flights to 15 different cities with airports to visit on our tour: Los Angeles (LAX), Denver (DEN), New York (JFK), Sydney (SYD), Seoul (ICN), Tokyo (HND), Osaka (ITM), Frankfurt (FRA), Amsterdam (AMS), London (LHR), Paris (CDG), Miami (MIA), Seattle (SEA), Chicago (ORD), and Dallas (DFW). To determine the cost of each flight, with one flight to every location on each day. The dataset is not entirely realistic, for example flights from the U.S. to Japan can take much longer than 1 day and flight record does not take into account yearly variations of flights with delays and cancellations, but creating such dynamic paradigms would likely have made the scope of the problem infeasible with the recent unpredictable collapses in software such as cloudstrike and AWS.

Formulation

Decision Variables:

X_i : set of binary indicator of whether each flight is chosen, where i is flight number

$Y_{c,d}$: set of binary indicator of whether there is concert in city c on date d

Parameters:

C: set of city candidates for tour

D: set of date candidates for tour, from May 1st to July 1st

F: set of flight candidates for tour

s: START_CITY is our starting city of the tour, Los Angeles

L=3: the maximum number of days we want to be able to stay in a city for

For any flight i in F:

origin(i): origin city of flight i

dest(i): destination city of flight i

t_i: departure date of flight i

p_i: ticket price of flight i

Code:

```
# Parameters
L = 3  # Maximum days staying in a city (including concert day)
START_CITY = "LAX"  # Fixed starting city
```

```
# Decision Variables
X = {}  # Flights
for i, row in flights.iterrows():
    f_id = row['flight_id']
    X[f_id] = m.addVar(vtype=GRB.BINARY, name=f"X_{f_id}")

Y = {}  # Concerts
for c in cities:
    for d in dates:
        Y[c, d] = m.addVar(vtype=GRB.BINARY, name=f"Y_{c}_{d}")
```

Objective: minimizing the price p_i of our flights x_i

$$Min_x z = \sum_{i=1}^{n} p_i x_i$$

Code:

```
m.setObjective(sum(X[row['flight_id']] * row['price_usd'] for i, row in flights.iterrows()),
GRB.MINIMIZE)
```

Constraint 1: Each city must have exactly one concert

$$\sum_{d \in D} y_{c,d} = 1, \forall c \in C$$

Code:

```
for c in cities:
    if c == START_CITY: # handled manually (concert before planning)
        continue
    m.addConstr(sum(Y[c, d] for d in dates) == 1, name=f"OneConcert_{c}")
```

Constraint 2: Each city must have exactly one incoming flight

Let $F_c = \{i \in F \mid dest(i) = c\}$, set of flights to city c

$$\sum_{i \in F_c} X_i = 1, \ \forall c \in C \setminus \{s\}$$

Code:

```
for c in cities:
    if c == START_CITY:
        continue # START_CITY incoming flight handled in later return-flight constraint
    relevant_flights = flights[flights['destination'] == c]['flight_id'].tolist()
    m.addConstr(sum(X[f] for f in relevant_flights) == 1, name=f"OneFlight_{c}")
```

Constraint 3: Describes linkage of ticket and concert destination

Let $W_i = \{d \in D \mid 1 \leq d - t_i \leq L - 1\}$, the concert dates for a flight i

$$X_i \leq \sum_{d \in W_i} Y_{c = dest(i),\, d}, \ \forall i \in F \text{ with dest(i)} \neq s$$

(1 to L-1 days after arrival)

Code:

```
# 3. Flight-concert linkage (arrival 1 day before to L-1 days after)
flight_allowed_dates = {}
for i, row in flights.iterrows():
    f_id = row['flight_id']
    c = row['destination']
    f_date = row['date_dt']

    if c == START_CITY:
        continue  # returning to start city doesn’t need a concert

    min_concert_date = f_date + pd.Timedelta(days=1)
    max_concert_date = f_date + pd.Timedelta(days=L-1)

    # getting all allowed dates for each concert
    for d in dates:
        if min_concert_date <= d <= max_concert_date:
            if f_id not in flight_allowed_dates:
                flight_allowed_dates[f_id] = []
            flight_allowed_dates[f_id].append(d)

for i, row in flights.iterrows():
    f_id = row['flight_id']
```

```
    c = row['destination']
    if c == START_CITY:
        continue
    allowed = flight_allowed_dates.get(f_id, [])
    if not allowed:
        m.addConstr(X[f_id] == 0, name=f"NoFeasibleDates_{f_id}")
    else:
        m.addConstr(X[f_id] <= sum(Y[c, d] for d in allowed),
name=f"ArrivalConcertWindow_{f_id}")
```

Constraint 4: No more than one concert per day globally

$$\sum_{c \in C} Y_{c,d} \leq 1, \ \forall d \in D$$

Code:

```
for d in dates:
    m.addConstr(sum(Y[c, d] for c in cities) <= 1, name=f"OneConcertPerDay_{d}")
```

Constraint 5: At most one flight per day

$$\text{Let } F_d = \{i \in F \mid t_i = d\}$$

$$\sum_{i \in F_d} X_i \leq 1, \ \forall d \in D$$

Code:

```
for d in dates:
    m.addConstr(sum(X[row['flight_id']] for i, row in flights.iterrows() if row['date_dt'] == d) <=
1, name=f"OneFlightPerDay_{d}")
```

Constraint 6: Flights must start from the previous city (more linkage)

$$X_i \leq Y_{origin(i),\, t_i - 1}$$

Code:

```
for i, row in flights.iterrows():
    f_id = row['flight_id']
    origin = row['origin']
    dest = row['destination']
    f_date = row['date_dt']

    # First flight must start from START_CITY
    if f_date == min(dates):
        if origin != START_CITY:
            m.addConstr(X[f_id] == 0, name=f"FirstFlightFromSTART_{f_id}")
        continue

    # Returning to START_CITY doesn't need origin check here
    if dest == START_CITY:
        continue
```

prev_day = f_date - pd.Timedelta(days=1)

m.addConstr(X[f_id] <= sum(Y[origin, d] for d in [prev_day] if d in dates),

name=f"FlightOriginCheck_{f_id}")

Constraint 7: Ensure one return flight to starting city

Let R = $\{i \in F \mid dest(i) = s\}$, the set of all flights with destination being STARTING_CITY

$$\sum_{i \in R} X_i = 1$$

$t_j X_j \leq \sum_{i \in R} t_i X_i, \ \forall j \in F$, where j is any other flight in the tour

Code:

return_flights = flights[flights['destination'] == START_CITY]['flight_id'].tolist()

m.addConstr(sum(X[f] for f in return_flights) == 1, name="ReturnToStartCity")

enforce return flight originates from previous concert city

for f_id in return_flights:

origin = flights.loc[flights['flight_id'] == f_id, 'origin'].values[0]

f_date = flights.loc[flights['flight_id'] == f_id, 'date_dt'].values[0]

prev_day = f_date - pd.Timedelta(days=1)

if prev_day in dates:

m.addConstr(X[f_id] <= sum(Y[origin, d] for d in [prev_day]),

name=f"ReturnOriginCheck_{f_id}")

Constraint 8: Binary Constraints

$$X_i \in \{0,1\}\ \forall i\ , Y_{c,d} \in \{0,1\}\ \forall c, d$$

Code:

Set in variable initialization step

Results and Post Optimality Analysis

Optimized Tour Schedule:

Concert in LAX on 2025-04-30

Concert in DEN on 2025-05-02

Concert in JFK on 2025-05-04

Concert in SYD on 2025-05-07

Concert in ICN on 2025-05-10

Concert in HND on 2025-05-13

Concert in ITM on 2025-05-15

Concert in FRA on 2025-05-18

Concert in AMS on 2025-05-21

Concert in LHR on 2025-05-24

Concert in CDG on 2025-05-27

Concert in MIA on 2025-05-29

Concert in SEA on 2025-06-01

Concert in ORD on 2025-06-03

Concert in DFW on 2025-06-05

Selected Flights:

Flight 81 from LAX to DEN on 2025-05-01 (Cost: 132)

Flight 1112 from DEN to JFK on 2025-05-03 (Cost: 144)

Flight 1924 from JFK to SYD on 2025-05-05 (Cost: 743)

Flight 3713 from SYD to ICN on 2025-05-08 (Cost: 800)

Flight 5093 from ICN to HND on 2025-05-11 (Cost: 139)

Flight 6454 from HND to ITM on 2025-05-14 (Cost: 116)

Flight 7425 from ITM to FRA on 2025-05-16 (Cost: 645)

Flight 8784 from FRA to AMS on 2025-05-19 (Cost: 114)

Flight 10221 from AMS to LHR on 2025-05-22 (Cost: 117)

Flight 11582 from LHR to CDG on 2025-05-25 (Cost: 126)

Flight 13050 from CDG to MIA on 2025-05-28 (Cost: 451)

Flight 13909 from MIA to SEA on 2025-05-30 (Cost: 118)

Flight 15357 from SEA to ORD on 2025-06-02 (Cost: 151)

Flight 16214 from ORD to DFW on 2025-06-04 (Cost: 151)

Flight 17207 from DFW to LAX on 2025-06-06 (Cost: 106)

The provided output above is the optimal ordering of concerts and optimal choices of flights to minimize the cost of arriving at each of the specified locations. The total resulting cost was $4,053. Gurobi took around 8 hours and 10 minutes to complete this program, exploring 228174 nodes with 56935632 simplex iterations. This is due to the size of our dataset and since the formulation of this problem is NP-hard as it shares traits with the travelling salesman problem. As a result, we dropped the idea of optimizing the cheapest airport linkages between each of the locations due to the lack of powerful hardware for optimization. Reduced Costs and Shadow Price analyses could not be explored here due to the nature of our decision variables being binary. For future exploration, consider stochastic effects such as flight cancellations, disruptions, and delays to boost accuracy and realism to our formulation and code. We can also add in the airport linkages of connecting flights which we omitted previously.

Appendix

Video Link: https://youtu.be/tmwSFNcOEiY

GitHub Link: https://github.com/kennyguki/Concert-Flight-Scheduling-Optimization

Task Allocation:

Ayaan Baig - 33.3333% of everything

Shan He - 33.3333% of everything

Kenneth Gutierrez King - 33.3333% of everything