Shortest Path to visit all 48 continental states

John Derr, Fall 2023, CS 524 Optimization @ UW - Madison

This was a project that I came up with to find something I have never seen before, for a private pilot, what is the shortest way to land in all 48 states. I have simplified this problem by mostly using class C airports to limit the total number of airports from over 5,211 public airports to just ~120. A class C airport is mid-size airport that service both general aviation and airline traffic with a Air Traffic Control Tower. I chose these airports as they are 122 of them, so there are plenty of possible paths to choose from.

First of all, several states that do not have class C airports. These states are Delaware, Maryland, Massachusetts, Minnesota, North Dakota, South Dakota, Utah, and Wyoming. I have decided that I still want to include these states. So I added several airports from each state that were reasonable in comparison to the other airports.

Secondly, there are some airports that were under 5 miles (8 km) apart from each other in the same state. Those have been removed to make the computation easier and the graph more readable.

Thirdly, some airports that were just not going to be used, like Key West and most of southern Florida and Texas, so they have been removed to speed up computation time. Bringing the total number of airports to 111. Finally, as part of the MTZ constraints to remove sub tours, I had to specify an airport that must be landed and departed from. I decided to call this my starting point, and as a true Badger, it is Madison, WI. The way you set the starting point is by having that point be the top line of your CSV.

In [1]:
```
useBig=True
```

In [2]:
```
def distance(lat1, lat2, lon1, lon2, km=False):

    # The math module contains a function named
    # radians which converts from degrees to radians.
    lon1 = math.radians(lon1)
    lon2 = math.radians(lon2)
    lat1 = math.radians(lat1)
    lat2 = math.radians(lat2)

    # Haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = math.sin(dlat / 2)**2 + math.cos(lat1) * math.cos(lat2) * math.sin(dlon / 2)**2

    c = 2 * math.asin(math.sqrt(a))

    r = 6371.0008 if km else 3958.7564

    # calculate the result
    return(c * r)

def plot_with_tour(tour, dist=0):
    city = allAirports.set_index('ICAO')[['ARP Latitude DD', 'ARP Longitude DD']].to_dict(orie
    tour.columns = ['from','to']
    us.plot(figsize=(20,20),color='white', edgecolor='black')
    # Use existing us geopandas dataframe as background
    for k, v in city.items():
        plt.scatter(v['ARP Longitude DD'],v['ARP Latitude DD'], c='black',alpha=0.5)
        plt.text(v['ARP Longitude DD'],v['ARP Latitude DD'], k, c='blue')
    for index, row in tour.iterrows():
        st = city[row['from']]
        ed = city[row['to']]
#         print(st['ARP Latitude DD'])
        plt.plot([st['ARP Longitude DD'], ed['ARP Longitude DD']], [st['ARP Latitude DD'], ed[
    plt.xlim(-130,-60)
    plt.ylim(23,53)
    plt.xlabel('Latitude')
    plt.ylabel('Longitude')
    if(dist != 0):
        plt.title(f'Tour Plot\nTour Length: {dist} miles')
    else:
        plt.title(f'Tour Plot')
    plt.savefig('test.png', bbox_inches='tight')
```

In [3]:
```
%load_ext gams.magic
m = gams.exchange_container
```

In [4]:

```
import geopandas, io, requests
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
import gams.transfer as gt
from functools import lru_cache

%matplotlib inline

@lru_cache(maxsize=None)
def get_world_df() -> geopandas.GeoDataFrame:
    natural_earth_url = ('https://www.naturalearthdata.com/'
                         'http//www.naturalearthdata.com/download/110m/cultural/ne_110m_admin_
    resp = requests.get(natural_earth_url, headers={"User-Agent": "XY"})
    df_world = geopandas.read_file(io.BytesIO(resp.content))
    return df_world

# load the world map and extract the map of United States of America
us = get_world_df()

filename = "airportsBIG.csv" if useBig else "airportsSMALL.csv"


allAirports = pd.read_csv(filename)

allAir = m.addSet('allAir', description='All Airports in the US that are considered', records=
m.addAlias("i",allAir)
m.addAlias("j",allAir)

numAirports = len(allAirports["ICAO"].to_list())

#Calculate distance matrix
dists = np.zeros((numAirports, numAirports))
for i in range(numAirports):
    for j in range(i+1, numAirports):
        dists[i,j] = distance(allAirports["ARP Latitude DD"][i],allAirports["ARP Latitude DD"]
        dists[j,i] = dists[i,j]

dist = m.addParameter("dist", [allAir,allAir], records=dists)


state_sets = {}
state_equations = {}
for state in allAirports['State Id'].unique():
    state_sets[state] = m.addSet(state, domain=allAir, records=allAirports.loc[allAirports['St
    m.addAlias(f"{state}1", state_sets[state])
    # Code to generate the equations used to solve:
#     print(f"  {state}eqIn\n  {state}eqOut\n  {state}eqSame")
#     print(state+f"eqIN..\n    sum(({state},i)$(not {state}1(i)), x(i,{state})) =e= 1;")
#     print(state+f"eqOut..\n    sum(({state},i)$(not {state}1(i)), x({state},i)) =e= 1;")
#     print(state+f"eqSame..\n    sum(({state},{state}1)$(not sameas({state},{state}1)), x({st
```

Before getting to my small version of the problem, there were several approaches I took to this. I found that optimizing the equations to be solved by Cplex difficult thus, the large problem takes over 6 hours to get to a relative error of 0.05. While I am running this on my laptop, I do not have the time nor ram to run this to my goal of 0.01 relative error.

On my first runs of the large dataset, I found that Cplex was generating a lot of edges in side each state, thus the Same equations were born, limiting the number of paths inside a state to 0. This is still guaranteed to be optimal as the shortest distance between 2 points is a line, not a triangle (even accounting for the curvature of the sphere). I also noticed that even if there were no internal edges, there were sub tours. I didn't know how this was possible at first, but after some consideration the IO equation was born. MTZ is designed for the TSP problem not set TSP, so I needed to add the IO equation to have the amount in and out paths for an airport be equal, thus eliminating sub tours.

So the approach I took, a TSP problem that requires that there be one landing and departure in each state, this landing and departure must be from the same airport. Then use MTZ constraints to eliminate sub tours.

Also, notice above how I decided to generate my equations, all the equations are the same for each state, but with GAMS, I am not able to combine them into one equation, so these print statements generated the 144 individual equations needed to solve.

Here is the small version of the model with only 3 states, making it much easier to read.

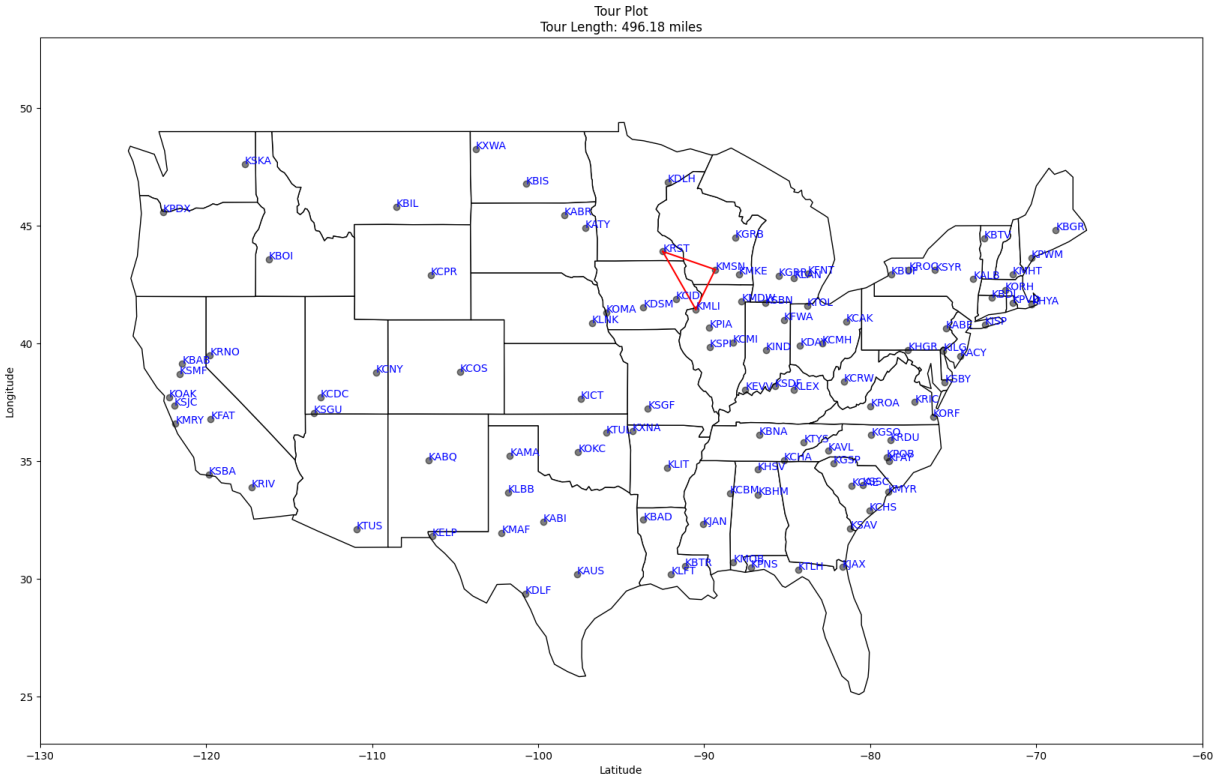
In [5]:
```
%%gams
option limrow=0,limcol=0,solprint=off;
positive variables u(i) MTZ var;
variable obj;
binary variable x(i,j);
equation mtz(i,j),defobj;

equation
ILeqIN
ILeqOUT
WIeqIN
WIeqOUT
MNeqIN
MNeqOUT
ILeqSame
WIeqSame
MNeqSame
IO(i);


ILeqIN..
    sum((IL,i)$(not IL1(i)), x(i,IL)) =e= 1;
ILeqOut..
    sum((IL,i)$(not IL1(i)), x(IL,i)) =e= 1;
ILeqSame..
    sum((IL,IL1)$(not sameas(IL,IL1)), x(IL,IL1)) =e= 0;
WIeqIN..
    sum((WI,i)$(not WI1(i)), x(i,WI)) =e= 1;
WIeqOut..
    sum((WI,i)$(not WI1(i)), x(WI,i)) =e= 1;
WIeqSame..
    sum((WI,WI1)$(not sameas(WI,WI1)), x(WI,WI1)) =e= 0;
MNeqIN..
    sum((MN, i)$(not MN1(i)), x(i,MN)) =e= 1;
MNeqOut..
    sum((MN, i)$(not MN1(i)), x(MN,i)) =e= 1;
MNeqSame..
    sum((MN,MN1)$(not sameas(MN,MN1)), x(MN,MN1)) =e= 0;
# Add equation stateing that if a location has an in, it must have an out
io(i)..
  sum(j$(not sameas(i,j)), x(i,j)) =g= sum(allAir$(not sameas(allAir,i)), x(allAir,i));


defobj..
  obj =e= sum((i,j), dist(i,j) * x(i,j));

mtz(i,j)$(ord(i) ne 1 and ord(j) ne 1)..
  u(i) - u(j) + 1 =L= (card(i) - 1) * (1 - x(i,j)) ;

# Used for MTZ
u.lo(i) = 2; u.up(i) = card(i);
u.fx(i)$(i.ord eq 1) = 1;


model test1 /mtz, defobj, io, MNeqSame,MNeqIN,MNeqOut,WIeqIn,WIeqSame,WIeqOut,ILeqSame,ILeqIN,
solve test1 using mip min obj;
set tour(i,i);
option tour:0:0:1;
tour(i,j) = no;
tour(i,j)$(x.l(i,j) > 0.01) = yes;
```

Out[5]:

	Solver Status	Model Status	Objective	#equ	#var	Model Type	Solver	Solver Time
0	Normal (1)	Optimal Global (1)	496.1823	12221	12431	MIP	CPLEX	0.25

In [6]:
```
t = m['tour']
tour = t.records[t.domain_labels].copy()
display(tour)
plot_with_tour(tour, round(m['obj'].records["level"][0], 2))
```

	i_0	i_1
0	KMSN	KRST
1	KMLI	KMSN
2	KRST	KMLI



This is exactly what I expected the optimal path to be for 3 states, a triangle. Now that I have this simple solution at hand, I was more confident with my equations ability to find the correct path with the larger dataset. The first time I ran this, I expected a full optimal solution in an hour. I was wrong. As I stated earlier, the 6 hour runs made this a "I hope this change works" type system.

In [9]:
```
%%gams
equations
  ALeqIn
  ALeqOut
  ALeqSame
  AZeqIn
  AZeqOut
  AZeqSame
  AReqIn
  AReqOut
  AReqSame
  CAeqIn
  CAeqOut
  CAeqSame
  COeqIn
  COeqOut
  COeqSame
  CTeqIn
  CTeqOut
  CTeqSame
  FLeqIn
  FLeqOut
  FLeqSame
  GAeqIn
  GAeqOut
  GAeqSame
  IDeqIn
  IDeqOut
  IDeqSame
  INeqIn
  INeqOut
  INeqSame
  IAeqIn
  IAeqOut
  IAeqSame
  KSeqIn
  KSeqOut
  KSeqSame
  KYeqIn
  KYeqOut
  KYeqSame
  LAeqIn
  LAeqOut
  LAeqSame
  MEeqIn
  MEeqOut
  MEeqSame
  MIeqIn
  MIeqOut
  MIeqSame
  MSeqIn
  MSeqOut
  MSeqSame
  MOeqIn
  MOeqOut
  MOeqSame
  MTeqIn
  MTeqOut
  MTeqSame
  NEeqIn
  NEeqOut
  NEeqSame
  NVeqIn
  NVeqOut
  NVeqSame
```

```
NHeqIn
NHeqOut
NHeqSame
NJeqIn
NJeqOut
NJeqSame
NMeqIn
NMeqOut
NMeqSame
NYeqIn
NYeqOut
NYeqSame
NCeqIn
NCeqOut
NCeqSame
OHeqIn
OHeqOut
OHeqSame
OKeqIn
OKeqOut
OKeqSame
OReqIn
OReqOut
OReqSame
PAeqIn
PAeqOut
PAeqSame
RIeqIn
RIeqOut
RIeqSame
SCeqIn
SCeqOut
SCeqSame
TNeqIn
TNeqOut
TNeqSame
TXeqIn
TXeqOut
TXeqSame
VTeqIn
VTeqOut
VTeqSame
VAeqIn
VAeqOut
VAeqSame
WAeqIn
WAeqOut
WAeqSame
WVeqIn
WVeqOut
WVeqSame
WYeqIn
WYeqOut
WYeqSame
UTeqIn
UTeqOut
UTeqSame
SDeqIn
SDeqOut
SDeqSame
NDeqIn
NDeqOut
NDeqSame
MAeqIn
MAeqOut
MAeqSame
```

```
  MAeqSame
  MDeqIn
  MDeqOut
  MDeqSame
  DEeqIn
  DEeqOut
  DEeqSame;

ALeqIN..
    sum((AL,i)$(not AL1(i)), x(i,AL)) =e= 1;
ALeqOut..
    sum((AL,i)$(not AL1(i)), x(AL,i)) =e= 1;
ALeqSame..
    sum((AL,AL1)$(not sameas(AL,AL1)), x(AL,AL1)) =e= 0;
AZeqIN..
    sum((AZ,i)$(not AZ1(i)), x(i,AZ)) =e= 1;
AZeqOut..
    sum((AZ,i)$(not AZ1(i)), x(AZ,i)) =e= 1;
AZeqSame..
    sum((AZ,AZ1)$(not sameas(AZ,AZ1)), x(AZ,AZ1)) =e= 0;
AReqIN..
    sum((AR,i)$(not AR1(i)), x(i,AR)) =e= 1;
AReqOut..
    sum((AR,i)$(not AR1(i)), x(AR,i)) =e= 1;
AReqSame..
    sum((AR,AR1)$(not sameas(AR,AR1)), x(AR,AR1)) =e= 0;
CAeqIN..
    sum((CA,i)$(not CA1(i)), x(i,CA)) =e= 1;
CAeqOut..
    sum((CA,i)$(not CA1(i)), x(CA,i)) =e= 1;
CAeqSame..
    sum((CA,CA1)$(not sameas(CA,CA1)), x(CA,CA1)) =e= 0;
COeqIN..
    sum((CO,i)$(not CO1(i)), x(i,CO)) =e= 1;
COeqOut..
    sum((CO,i)$(not CO1(i)), x(CO,i)) =e= 1;
COeqSame..
    sum((CO,CO1)$(not sameas(CO,CO1)), x(CO,CO1)) =e= 0;
CTeqIN..
    sum((CT,i)$(not CT1(i)), x(i,CT)) =e= 1;
CTeqOut..
    sum((CT,i)$(not CT1(i)), x(CT,i)) =e= 1;
CTeqSame..
    sum((CT,CT1)$(not sameas(CT,CT1)), x(CT,CT1)) =e= 0;
FLeqIN..
    sum((FL,i)$(not FL1(i)), x(i,FL)) =e= 1;
FLeqOut..
    sum((FL,i)$(not FL1(i)), x(FL,i)) =e= 1;
FLeqSame..
    sum((FL,FL1)$(not sameas(FL,FL1)), x(FL,FL1)) =e= 0;
GAeqIN..
    sum((GA,i)$(not GA1(i)), x(i,GA)) =e= 1;
GAeqOut..
    sum((GA,i)$(not GA1(i)), x(GA,i)) =e= 1;
GAeqSame..
    sum((GA,GA1)$(not sameas(GA,GA1)), x(GA,GA1)) =e= 0;
IDeqIN..
    sum((ID,i)$(not ID1(i)), x(i,ID)) =e= 1;
IDeqOut..
    sum((ID,i)$(not ID1(i)), x(ID,i)) =e= 1;
IDeqSame..
    sum((ID,ID1)$(not sameas(ID,ID1)), x(ID,ID1)) =e= 0;
INeqIN..
    sum((IN,i)$(not IN1(i)), x(i,IN)) =e= 1;
INeqOut..
    sum((IN,i)$(not IN1(i)), x(IN,i)) =e= 1;
```

```
INeqSame..
    sum((IN,IN1)$(not sameas(IN,IN1)), x(IN,IN1)) =e= 0;
IAeqIN..
    sum((IA,i)$(not IA1(i)), x(i,IA)) =e= 1;
IAeqOut..
    sum((IA,i)$(not IA1(i)), x(IA,i)) =e= 1;
IAeqSame..
    sum((IA,IA1)$(not sameas(IA,IA1)), x(IA,IA1)) =e= 0;
KSeqIN..
    sum((KS,i)$(not KS1(i)), x(i,KS)) =e= 1;
KSeqOut..
    sum((KS,i)$(not KS1(i)), x(KS,i)) =e= 1;
KSeqSame..
    sum((KS,KS1)$(not sameas(KS,KS1)), x(KS,KS1)) =e= 0;
KYeqIN..
    sum((KY,i)$(not KY1(i)), x(i,KY)) =e= 1;
KYeqOut..
    sum((KY,i)$(not KY1(i)), x(KY,i)) =e= 1;
KYeqSame..
    sum((KY,KY1)$(not sameas(KY,KY1)), x(KY,KY1)) =e= 0;
LAeqIN..
    sum((LA,i)$(not LA1(i)), x(i,LA)) =e= 1;
LAeqOut..
    sum((LA,i)$(not LA1(i)), x(LA,i)) =e= 1;
LAeqSame..
    sum((LA,LA1)$(not sameas(LA,LA1)), x(LA,LA1)) =e= 0;
MEeqIN..
    sum((ME,i)$(not ME1(i)), x(i,ME)) =e= 1;
MEeqOut..
    sum((ME,i)$(not ME1(i)), x(ME,i)) =e= 1;
MEeqSame..
    sum((ME,ME1)$(not sameas(ME,ME1)), x(ME,ME1)) =e= 0;
MIeqIN..
    sum((MI,i)$(not MI1(i)), x(i,MI)) =e= 1;
MIeqOut..
    sum((MI,i)$(not MI1(i)), x(MI,i)) =e= 1;
MIeqSame..
    sum((MI,MI1)$(not sameas(MI,MI1)), x(MI,MI1)) =e= 0;
MSeqIN..
    sum((MS,i)$(not MS1(i)), x(i,MS)) =e= 1;
MSeqOut..
    sum((MS,i)$(not MS1(i)), x(MS,i)) =e= 1;
MSeqSame..
    sum((MS,MS1)$(not sameas(MS,MS1)), x(MS,MS1)) =e= 0;
MOeqIN..
    sum((MO,i)$(not MO1(i)), x(i,MO)) =e= 1;
MOeqOut..
    sum((MO,i)$(not MO1(i)), x(MO,i)) =e= 1;
MOeqSame..
    sum((MO,MO1)$(not sameas(MO,MO1)), x(MO,MO1)) =e= 0;
MTeqIN..
    sum((MT,i)$(not MT1(i)), x(i,MT)) =e= 1;
MTeqOut..
    sum((MT,i)$(not MT1(i)), x(MT,i)) =e= 1;
MTeqSame..
    sum((MT,MT1)$(not sameas(MT,MT1)), x(MT,MT1)) =e= 0;
NEeqIN..
    sum((NE,i)$(not NE1(i)), x(i,NE)) =e= 1;
NEeqOut..
    sum((NE,i)$(not NE1(i)), x(NE,i)) =e= 1;
NEeqSame..
    sum((NE,NE1)$(not sameas(NE,NE1)), x(NE,NE1)) =e= 0;
NVeqIN..
    sum((NV,i)$(not NV1(i)), x(i,NV)) =e= 1;
NVeqOut..
    sum((NV,i)$(not NV1(i)), x(NV,i)) =e= 1;
```

```
NVeqSame..
    sum((NV,NV1)$(not sameas(NV,NV1)), x(NV,NV1)) =e= 0;
NHeqIN..
    sum((NH,i)$(not NH1(i)), x(i,NH)) =e= 1;
NHeqOut..
    sum((NH,i)$(not NH1(i)), x(NH,i)) =e= 1;
NHeqSame..
    sum((NH,NH1)$(not sameas(NH,NH1)), x(NH,NH1)) =e= 0;
NJeqIN..
    sum((NJ,i)$(not NJ1(i)), x(i,NJ)) =e= 1;
NJeqOut..
    sum((NJ,i)$(not NJ1(i)), x(NJ,i)) =e= 1;
NJeqSame..
    sum((NJ,NJ1)$(not sameas(NJ,NJ1)), x(NJ,NJ1)) =e= 0;
NMeqIN..
    sum((NM,i)$(not NM1(i)), x(i,NM)) =e= 1;
NMeqOut..
    sum((NM,i)$(not NM1(i)), x(NM,i)) =e= 1;
NMeqSame..
    sum((NM,NM1)$(not sameas(NM,NM1)), x(NM,NM1)) =e= 0;
NYeqIN..
    sum((NY,i)$(not NY1(i)), x(i,NY)) =e= 1;
NYeqOut..
    sum((NY,i)$(not NY1(i)), x(NY,i)) =e= 1;
NYeqSame..
    sum((NY,NY1)$(not sameas(NY,NY1)), x(NY,NY1)) =e= 0;
NCeqIN..
    sum((NC,i)$(not NC1(i)), x(i,NC)) =e= 1;
NCeqOut..
    sum((NC,i)$(not NC1(i)), x(NC,i)) =e= 1;
NCeqSame..
    sum((NC,NC1)$(not sameas(NC,NC1)), x(NC,NC1)) =e= 0;
OHeqIN..
    sum((OH,i)$(not OH1(i)), x(i,OH)) =e= 1;
OHeqOut..
    sum((OH,i)$(not OH1(i)), x(OH,i)) =e= 1;
OHeqSame..
    sum((OH,OH1)$(not sameas(OH,OH1)), x(OH,OH1)) =e= 0;
OKeqIN..
    sum((OK,i)$(not OK1(i)), x(i,OK)) =e= 1;
OKeqOut..
    sum((OK,i)$(not OK1(i)), x(OK,i)) =e= 1;
OKeqSame..
    sum((OK,OK1)$(not sameas(OK,OK1)), x(OK,OK1)) =e= 0;
OReqIN..
    sum((OR,i)$(not OR1(i)), x(i,OR)) =e= 1;
OReqOut..
    sum((OR,i)$(not OR1(i)), x(OR,i)) =e= 1;
OReqSame..
    sum((OR,OR1)$(not sameas(OR,OR1)), x(OR,OR1)) =e= 0;
PAeqIN..
    sum((PA,i)$(not PA1(i)), x(i,PA)) =e= 1;
PAeqOut..
    sum((PA,i)$(not PA1(i)), x(PA,i)) =e= 1;
PAeqSame..
    sum((PA,PA1)$(not sameas(PA,PA1)), x(PA,PA1)) =e= 0;
RIeqIN..
    sum((RI,i)$(not RI1(i)), x(i,RI)) =e= 1;
RIeqOut..
    sum((RI,i)$(not RI1(i)), x(RI,i)) =e= 1;
RIeqSame..
    sum((RI,RI1)$(not sameas(RI,RI1)), x(RI,RI1)) =e= 0;
SCeqIN..
    sum((SC,i)$(not SC1(i)), x(i,SC)) =e= 1;
SCeqOut..
    sum((SC,i)$(not SC1(i)), x(SC,i)) =e= 1;
```

```
    sum((SC,i)$(not SC1(i)), x(SC,i)) =e= 1;
SCeqSame..
    sum((SC,SC1)$(not sameas(SC,SC1)), x(SC,SC1)) =e= 0;
TNeqIN..
    sum((TN,i)$(not TN1(i)), x(i,TN)) =e= 1;
TNeqOut..
    sum((TN,i)$(not TN1(i)), x(TN,i)) =e= 1;
TNeqSame..
    sum((TN,TN1)$(not sameas(TN,TN1)), x(TN,TN1)) =e= 0;
TXeqIN..
    sum((TX,i)$(not TX1(i)), x(i,TX)) =e= 1;
TXeqOut..
    sum((TX,i)$(not TX1(i)), x(TX,i)) =e= 1;
TXeqSame..
    sum((TX,TX1)$(not sameas(TX,TX1)), x(TX,TX1)) =e= 0;
VTeqIN..
    sum((VT,i)$(not VT1(i)), x(i,VT)) =e= 1;
VTeqOut..
    sum((VT,i)$(not VT1(i)), x(VT,i)) =e= 1;
VTeqSame..
    sum((VT,VT1)$(not sameas(VT,VT1)), x(VT,VT1)) =e= 0;
VAeqIN..
    sum((VA,i)$(not VA1(i)), x(i,VA)) =e= 1;
VAeqOut..
    sum((VA,i)$(not VA1(i)), x(VA,i)) =e= 1;
VAeqSame..
    sum((VA,VA1)$(not sameas(VA,VA1)), x(VA,VA1)) =e= 0;
WAeqIN..
    sum((WA,i)$(not WA1(i)), x(i,WA)) =e= 1;
WAeqOut..
    sum((WA,i)$(not WA1(i)), x(WA,i)) =e= 1;
WAeqSame..
    sum((WA,WA1)$(not sameas(WA,WA1)), x(WA,WA1)) =e= 0;
WVeqIN..
    sum((WV,i)$(not WV1(i)), x(i,WV)) =e= 1;
WVeqOut..
    sum((WV,i)$(not WV1(i)), x(WV,i)) =e= 1;
WVeqSame..
    sum((WV,WV1)$(not sameas(WV,WV1)), x(WV,WV1)) =e= 0;
WYeqIN..
    sum((WY,i)$(not WY1(i)), x(i,WY)) =e= 1;
WYeqOut..
    sum((WY,i)$(not WY1(i)), x(WY,i)) =e= 1;
WYeqSame..
    sum((WY,WY1)$(not sameas(WY,WY1)), x(WY,WY1)) =e= 0;
UTeqIN..
    sum((UT,i)$(not UT1(i)), x(i,UT)) =e= 1;
UTeqOut..
    sum((UT,i)$(not UT1(i)), x(UT,i)) =e= 1;
UTeqSame..
    sum((UT,UT1)$(not sameas(UT,UT1)), x(UT,UT1)) =e= 0;
SDeqIN..
    sum((SD,i)$(not SD1(i)), x(i,SD)) =e= 1;
SDeqOut..
    sum((SD,i)$(not SD1(i)), x(SD,i)) =e= 1;
SDeqSame..
    sum((SD,SD1)$(not sameas(SD,SD1)), x(SD,SD1)) =e= 0;
NDeqIN..
    sum((ND,i)$(not ND1(i)), x(i,ND)) =e= 1;
NDeqOut..
    sum((ND,i)$(not ND1(i)), x(ND,i)) =e= 1;
NDeqSame..
    sum((ND,ND1)$(not sameas(ND,ND1)), x(ND,ND1)) =e= 0;
MAeqIN..
    sum((MA,i)$(not MA1(i)), x(i,MA)) =e= 1;
MAeqOut..
```

```
    sum((MA,i)$(not MA1(i)), x(MA,i)) =e= 1;
MAeqSame..
    sum((MA,MA1)$(not sameas(MA,MA1)), x(MA,MA1)) =e= 0;
MDeqIN..
    sum((MD,i)$(not MD1(i)), x(i,MD)) =e= 1;
MDeqOut..
    sum((MD,i)$(not MD1(i)), x(MD,i)) =e= 1;
MDeqSame..
    sum((MD,MD1)$(not sameas(MD,MD1)), x(MD,MD1)) =e= 0;
DEeqIN..
    sum((DE,i)$(not DE1(i)), x(i,DE)) =e= 1;
DEeqOut..
    sum((DE,i)$(not DE1(i)), x(DE,i)) =e= 1;
DEeqSame..
    sum((DE,DE1)$(not sameas(DE,DE1)), x(DE,DE1)) =e= 0;

x.l(i,j) = 0;

* A valid solution found on an optcr = 0.1 to give cplex a valid solution to start from

x.l("KMSN","KMDW") = 1;x.l("KMDW","KSBN") = 1;x.l("KSBN","KLAN") = 1;x.l("KLAN","KTOL") = 1;
x.l("KTOL","KCRW") = 1;x.l("KCRW","KRIC") = 1;x.l("KRIC","KSBY") = 1;x.l("KSBY","KACY") = 1;
x.l("KACY","KILG") = 1;x.l("KILG","KABE") = 1;x.l("KABE","KBDL") = 1;x.l("KBDL","KPVD") = 1;
x.l("KPVD","KORH") = 1;x.l("KORH","KMHT") = 1;x.l("KMHT","KPWM") = 1;x.l("KPWM","KBTV") = 1;
x.l("KBTV","KALB") = 1;x.l("KALB","KLEX") = 1;x.l("KLEX","KTYS") = 1;x.l("KTYS","KAVL") = 1;
x.l("KAVL","KGSP") = 1;x.l("KGSP","KSAV") = 1;x.l("KSAV","KPNS") = 1;x.l("KPNS","KMOB") = 1;
x.l("KMOB","KBTR") = 1;x.l("KBTR","KJAN") = 1;x.l("KJAN","KLIT") = 1;x.l("KLIT","KSGF") = 1;
x.l("KSGF","KTUL") = 1;x.l("KTUL","KICT") = 1;x.l("KICT","KCOS") = 1;x.l("KCOS","KABQ") = 1;
x.l("KABQ","KELP") = 1;x.l("KELP","KTUS") = 1;x.l("KTUS","KSGU") = 1;x.l("KSGU","KRNO") = 1;
x.l("KRNO","KBAB") = 1;x.l("KBAB","KPDX") = 1;x.l("KPDX","KSKA") = 1;x.l("KSKA","KBOI") = 1;
x.l("KBOI","KBIL") = 1;x.l("KBIL","KCPR") = 1;x.l("KCPR","KBIS") = 1;x.l("KBIS","KABR") = 1;
x.l("KABR","KOMA") = 1;x.l("KOMA","KDSM") = 1;x.l("KDSM","KRST") = 1;x.l("KRST","KMSN") = 1;

model full /all/;
option threads = 16;
option Optcr = 0.05;
solve full using mip min obj;
set tour(i,i);
option tour:0:0:1;
tour(i,j) = no;
tour(i,j)$(x.l(i,j) > 0.01) = yes;
```

Out[9]:

	Solver Status	Model Status	Objective	#equ	#var	Model Type	Solver	Solver Time
0	Normal (1)	Integer (8)	9237.3514	12336	12431	MIP	CPLEX	19805.328

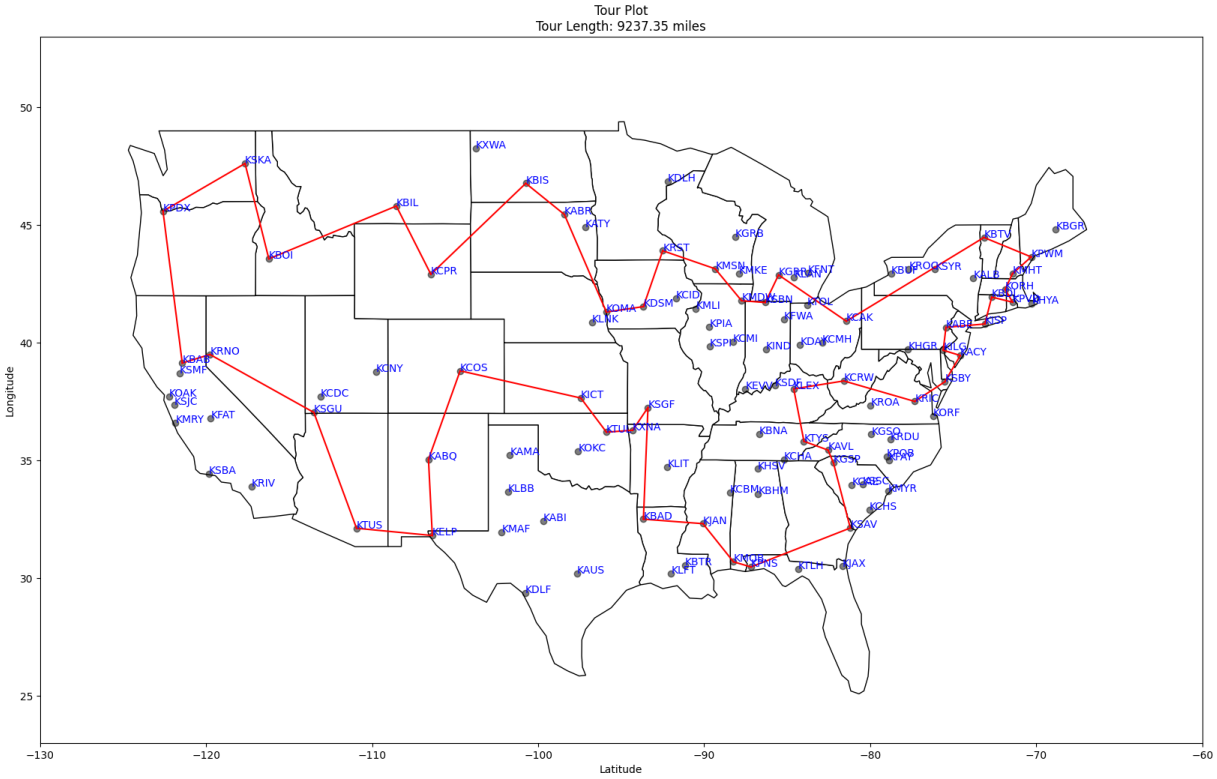
And 10,000 miles is a what my internal estimate was, so this passes what I was expecting. While I know this still has a 5% potential relative error, this is needed to make the solve time reasonable.

In [10]:

```
t = m['tour']
tour = t.records[t.domain_labels].copy()
display(tour)
plot_with_tour(tour, round(m['obj'].records["level"][0], 2))
```

	i_0	i_1
0	KMSN	KMDW
1	KMOB	KJAN
2	KTUS	KSGU
3	KXNA	KTUL
4	KBAB	KPDX
5	KCOS	KABQ
6	KBDL	KISP
7	KPNS	KMOB
8	KSAV	KPNS
9	KBOI	KBIL
10	KMDW	KSBN
11	KSBN	KGRR
12	KDSM	KRST
13	KICT	KCOS
14	KLEX	KTYS
15	KBAD	KSGF
16	KPWM	KMHT
17	KGRR	KCAK
18	KJAN	KBAD
19	KSGF	KXNA
20	KBIL	KCPR
21	KOMA	KDSM
22	KRNO	KBAB
23	KMHT	KORH
24	KACY	KSBY
25	KABQ	KELP
26	KISP	KABE
27	KAVL	KGSP
28	KCAK	KBTV
29	KTUL	KICT
30	KPDX	KSKA
31	KABE	KILG
32	KPVD	KBDL
33	KGSP	KSAV
34	KTYS	KAVL
35	KELP	KTUS
36	KBTV	KPWM

	i_0	i_1
37	KRIC	KCRW
38	KSKA	KBOI
39	KCRW	KLEX
40	KCPR	KBIS
41	KSGU	KRNO
42	KABR	KOMA
43	KBIS	KABR
44	KRST	KMSN
45	KORH	KPVD
46	KSBY	KRIC
47	KILG	KACY



Looking at this graph, this is approximately what I expected the path to look like, but I did not consider the KELP airport down in Texas. When I first saw this graph, I thought I messed up my equations because Texas was missed, but the optimization system did its job and found that KELP was not far from the route KABQ to KTUS.

Overall, this is the optimal path to visit all lower 48 states. While there is certainly limitations to this approach it is still solvable.

Extensions

Based on my reading, there is few papers written about the set TSP problem, thus I was going at this on my own. But based on the large variety of methods in traditional programming used to solve the TSP problem far faster, I believe that there is some better algorithms that can be used to solve this problem. With the better algorithms, it would be possible to add more airports to each state, thus generating a more realistic path.

This model does not consider aircraft fuel, but this was done on purpose. Each airport in the set I used has fuel available for sale, and by the nature of the TSP problem, there are not many long legs in the path. This is a possible extension but I feel that it would not cause much change to the route generated.

A final extension would be to mountains and lakes. Most private pilots have a hard time flying over the Rocky Mountains and do not like flying over the Great Lakes. It would be possible to add a soft constraint to keep the route from going over the worst of the mountains and lakes.

In [ ]:
```
%gams_cleanup --closedown
```

In [ ]:
```
```