CS/ECE/ISyE 524 — Introduction to Optimisation — Summer 2020

Siting Electric Vehicle Fast Charging Stations in Wisconsin

Winnie Chan

Table of Contents

1 Introduction

Concerns about electric vehicle (EV) driving range and the availability of charging stations present major obstacles to EV market penetration. In Wisconsin alone, the number of battery EVs (BEVs) barely surpassed 5,000 in 2019 – a trifling number dwarfed, for instance, by the more than 300,000 BEVs in California [1]. This presents a significant opportunity for more sustainable forms of transportation to be encouraged in the coming years.

Compared to Level 1 and Level 2 AC chargers, which are the chargers of choice for EVs at home and in workplaces, DC fast chargers are commonly deployed at public charging stations, relying on 3-phase high voltage electricity to supply powers of 50kW or more. With this, they can add anywhere from 150 to 1000 miles of range per hour charged, instead of the 3-6 miles or 10-60 miles per hour Level 1 and Level 2 AC charging respectively provide. Although AC charging is essential for the overnight recharging of EVs that are parked at home or are part of business fleets, it cannot

compete with DC fast charging in terms of convenience and efficiency, which are critical factors influencing how willing the public is to forego traditional internal combustion engine vehicles for EVs.

To address the needs of drivers who wish to traverse large distances across the state, but would rather avoid the hours-long charging times associated with the standard Level 1 and Level 2 AC charging stations, DC fast chargers will have to be installed at strategic locations in Wisconsin. Nevertheless, a poor appetite for higher risk and longer payback periods has generally discouraged private investors from backing fast charging, and DC infrastructure deployment has primarily been under the purview of governments, and in a stage of infancy [2]. Due to the present inaccessibility of fast chargers in Wisconsin, EVs are sometimes forced to take circuitous routes that pass by charging stations, instead of travelling more sensible, shorter paths between their origin and destination [3]. With $10 million in funds set aside by the Volkswagen Diesel Emissions Environmental Mitigation Trust to fund EV charging station grants earlier in 2020 [4], the Wisconsin state government finally has the financial means to address this situation.

Our project aims to propose where fast charging stations can be economically sited in Wisconsin to enable EVs to travel the shortest path on major transport arteries between major cities in and around the state. Our approach first establishes the shortest-path routes between pairs of origins and destinations (hereafter "OD pairs"), based on a model of the Wisconsin road transport network drawn up from the careful studying of maps. Following which, we build and test two optimisation models that determine where best to locate fast charging stations. The first one assumes an infinite charging station budget in order to cover all routes, while the second one optimises a limited budget to cover as many routes as possible.

Due to our approach to the problem, numerous assumptions had to be applied. Notably, charging station demand is ignored: we assume that stations have unlimited capacity, such that a single station at any given node is able to service all EV users at that site. Moreover, factors related to battery recharging, like charging duration and electrical grid supply, are neglected. Our analysis also assumes independence from considering the role of Level 1 and 2 AC chargers.

Further assumptions that are specific to our mathematical models are specified in the section of our report examining the Station Locator problem.

2 Data Generation

OD nodes were chosen by identifying urban centers with sizeable populations in or around Wisconsin. They are located along major highways/interstates, and often serve as important transit points. Another consideration is that OD candidate nodes are not too close together - for instance, creating nodes for both Milwaukee and Waukesha would be of limited utility given EVs' current driving ranges that far exceed the 20 miles separating Waukesha from Milwaukee Bay.

Transit nodes are non-OD locations. Their inclusion is necessary in cases where the shortest path between adjacent OD nodes has a distance greater than the driving range of an EV. As an illustration, the distance between Eau Claire and Madison is approximately 180 miles. Since there are no OD nodes between them on the transport network, an EV with a range of 150 miles will be unable to travel from one to the other without the aid of a charging station located at either the Portage or Tomah intermediate nodes.

A table specifying the set of nodes and the relative installation costs assigned to them is shown

below.

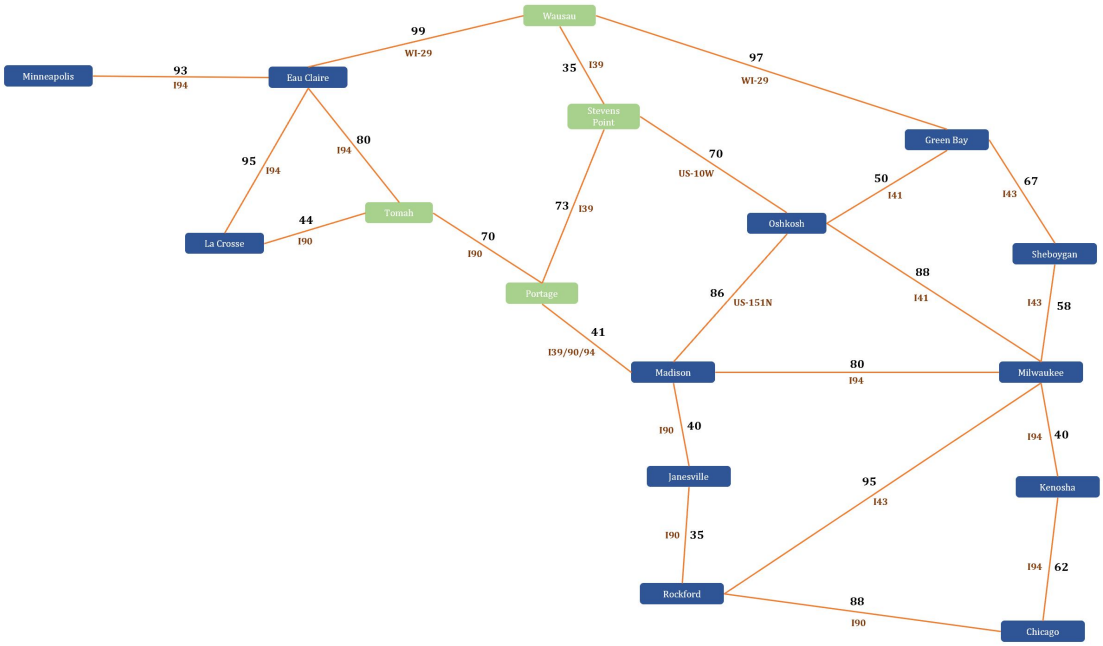
Abbreviation	Location	Node Type	Cost Coefficients
EAU	Eau Claire (WI)	OD	1.4
GRB	Green Bay (WI)	OD	1.2
JVL	Janesville (WI)	OD	1.4
KNS	Kenosha (WI)	OD	1.2
LCS	La Crosse (WI)	OD	1.4
MKE	Milwaukee (WI)	OD	0.85
MSN	Madison (WI)	OD	1.0
OSH	Oshkosh (WI)	OD	1.4
PTG	Portage (WI)	Transit	2.0
SBG	Sheboygan (WI)	OD	1.4
STP	Stevens Point (WI)	Transit	1.8
TOM	Tomah (WI)	Transit	2.0
WSA	Wausau (WI)	Transit	1.8
CHI	Chicago (IL)	OD	0.65
RFD	Rockford (IL)	OD	1.2
MPS	Minneapolis (MN)	OD	0.65

Edges each consist of two uni-directional arcs, representing how travel between any two nodes can occur in both directions. In the transport network constructed, the edges represent the I39, I41, I43, I90, I94, US-10W, and WI-29. Distance data for each edge was derived from Google Maps estimates for private vehicle trips.

Each node's infrastructure and population were studied to develop back-of-the-envelope estimates of how much it costs to purchase and install a fast charging station there compared to doing so in Madison. The resultant **cost coefficients** generated are relative instead of absolute values. This decision was made in light of how non-home charging hardware costs have been decreasing at about 3% annually [5], such that monetary values produced by more fine-grained cost analyses are quickly made redundant.

In 2015, a survey for the US DOE reported fast charging station installation costs ranged from as low as $8,500 to upwards of $50,000 [6]. This wide variation reflects the costly electrical service upgrades needed at more rural locations to ensure the provision of sufficient electricity to charging sites. In contrast, major metropolitan areas like Chicago or Milwaukee can exploit existing electrical service infrastructure to drive down costs. These considerations were taken into account in creating our cost weightings for each node.

The Wisconsin transport network is shown below, with numbers specifying distance in miles and alphanumeric labels specifying highway or interstate. OD and transit nodes are coloured navy and green respectively.



[1]:
```
### Data ###

# 12 origin/destination nodes
ODnodes = [:MPS, :EAU, :LCS, :MKE, :MSN, :GRB, :SBG, :OSH, :KNS, :JVL, :RFD, :
 ↪CHI]
# all nodes, including those representing interstate junctions and smaller cities
nodes = [:PTG, :TOM, :STP, :WSA]
nodes = vcat(ODnodes, nodes) # combines the two arrays of nodes

# unique set of (unidirectional) edge pairs
unidir_arcs = [(:MPS, :EAU), (:EAU, :LCS), (:EAU, :TOM), (:EAU, :WSA), (:LCS, :
 ↪TOM), (:TOM, :PTG), (:WSA, :STP),
    (:STP, :PTG), (:PTG, :MSN), (:MSN, :JVL), (:JVL, :RFD), (:WSA, :GRB), (:GRB,␣
 ↪:SBG), (:SBG, :MKE), (:MKE, :KNS),
    (:KNS, :CHI), (:STP, :OSH), (:OSH, :MSN), (:GRB, :OSH), (:OSH, :MKE), (:MSN,␣
 ↪:MKE), (:MKE, :RFD), (:RFD, :CHI)]
# add edge pairs in the reverse direction to complete the set
arcs = [unidir_arcs; [(j, i) for (i, j) in unidir_arcs]]

# distances on each edge
dists = [93 95 80 99 44 70 35 73 41 40 35 97 67 58 40 62 70 86 50 88 80 95 88]'␣
 ↪# corresponds to unidir_arcs
dists = vcat(dists, dists) # duplicate the distances so the array corresponds to␣
 ↪arcs
costs = Dict(zip(arcs, dists))
```

```
# relative installation costs
weights = Dict(zip(nodes, [0.65 1.4 1.4 0.85 1 1.2 1.4 1.4 1.2 1.4  1.2 0.65 2 2␣
 ↪1.8 1.8]));
```

3 Modelling

3.1 The Minimum Cost Network Flow problem

To simulate long distance travel, identification of the routes a driver will take between the different OD combinations is necessary. We implement a MCNF linear program to do so. It is solved 66 times, for each of the OD pairs, and returns which edges are traversed to give the shortest-path route of every OD pair.

The solutions of the MCNF model are subsequently used as parameters for our main Station Locator model.

3.1.1 MCNF mathematical model

Data
N - set of nodes
E - set of arcs
b_i - demand/supply of node $i \in N$
c_{ij} - cost of travelling on arc $i, j \in E$ (i.e. distance between node i and node j)
u_{ij} - upper bound on arc $i, j \in E$; set to 1
l_{ij} - lower bound on arc $i, j \in E$; set to 0
O - route origin node
D - route destination node

Variables
x_{ij} - total flow on arc $i, j \in E$

Model

$$\begin{aligned}
\underset{x}{\text{minimise}} \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} & & \\
\text{subject to:} \quad & l_{ij} \leq x_{ij} \leq u_{ij} & \forall (i,j) \in E & \quad (1)\\
& \sum_{j\in N} x_{kj} - \sum_{i\in N} x_{ik} = b_k & \forall k \in N & \quad (2)\\
& \sum_{i\in N} x_{ik} = 0 & \forall k \in \{O, D\} & \quad (3)
\end{aligned}$$

The objective of this LP is to find a minimum-distance path between a given origin and destination pair.

Constraint (1) defines the capacity on each arc to be either 0 or 1. Constraint (2) enforces flow balance into and out of each node, for which b_k is 1 for the source node O, is -1 for the sink node D, and is 0 for all other intermediate nodes. Constraint (3) prevents the formation of 'subtours',

in which flow exits from either O or D to the nearest adjacent node, only to loop back around and re-enter O or D.

3.1.2 MCNF implementation

[2]:
```
# ---------- Minimum Cost Network Flow model ---------- #
using JuMP, Clp

function shortest_path(O, D) # function takes in origin and destination nodes as␣
↪arguments
    m1 = Model(Clp.Optimizer)
    set_optimizer_attribute(m1, "LogLevel", 0)

    @variable(m1, x[arcs] >= 0 ) # binary variable representing whether a given␣
↪arc is traversed

    for i in nodes
        # if node is neither source nor sink
        if i != O && i != D
            @constraint(m1, sum(x[j] for j in arcs if j[1] == i) == sum(x[j] for␣
↪j in arcs if j[2] == i) )
        # if node is origin node
        elseif i == O
            @constraint(m1, sum(x[j] for j in arcs if j[1] == O) == 1 )
            @constraint(m1, sum(x[j] for j in arcs if j[2] == O) == 0 ) #␣
↪prevent backflow on reverse arc
        # if node is destination node
        else
            @constraint(m1, sum(x[j] for j in arcs if j[2] == D) == 1 )
            @constraint(m1, sum(x[j] for j in arcs if j[1] == D) == 0 ) #␣
↪prevent backflow on reverse arc
        end
    end

    @objective(m1, Min, sum(costs[i]*x[i] for i in arcs) )

    optimize!(m1)

    route = []
    dist = 0 # variable to record distance travelled on route
    for i in arcs
        if value(x[i]) == 1 # if an arc is traversed, add it to the OD route
            push!(route, i)
            dist += costs[i]
        end
    end
```

```
    return(route, dist)
end;
```

We now iteratively call our *shortest_path* function, using different pairs of origin and destination nodes each time. The solution of each instance is stored in a dictionary, *route_record*, which keeps both the arcs and the nodes involved in each route.

[3]:
```
### parameter collection for use in subsequent models ###
# initialisation
route_record = Dict() # create empty dictionary which records the routes of each␣
 ↪OD pair

using NamedArrays
n = length(ODnodes)
dists_record = NamedArray(zeros(n, n), (ODnodes, ODnodes), ("Origin",␣
 ↪"Destination"))


# since the route from A to B will be identical to the route from B to A, we␣
 ↪avoid redundancy and use the following:
for i in 1:length(ODnodes)
    for j in i+1:length(ODnodes)
        (current_route, current_dist) = shortest_path(ODnodes[i], ODnodes[j])

        # create a unique key for each entry in route_record
        key = string(ODnodes[i], "-", ODnodes[j])
        # the corresponding value for each key will be a list containing a route␣
 ↪(in terms of arcs) and its origin node
        route_record[key] = [current_route, ODnodes[i]]
        for j in current_route # add all the remaining nodes along the route to␣
 ↪route_record
            push!(route_record[key], j[2])
        end

        dists_record[i,j] = current_dist
        dists_record[j,i] = current_dist # distance from A to B is constant␣
 ↪regardless of direction travelled

    end
end
```

Since Minneapolis and Chicago represent the northwestern-most and southeastern-most points of our Wisconsin transport network, effectively serving as its 'bookends', we expect travel from one to the other to be along our longest route generated.

Let's confirm our code works, and see what the Minneapolis-Chicago entry in *route_records* shows.

[4]:
```
route_record["MPS-CHI"]
```

```
[4]: 9-element Array{Any,1}:
      Any[(:MPS, :EAU), (:EAU, :TOM), (:TOM, :PTG), (:PTG, :MSN), (:MSN, :JVL),
     (:JVL, :RFD), (:RFD, :CHI)]
      :MPS
      :EAU
      :TOM
      :PTG
      :MSN
      :JVL
      :RFD
      :CHI
```

The first element is a route consisting of the arcs which need to be traversed, and the subsequent elements are the nodes we pass through on that route.

Interestingly, Google Maps suggests an identical route when travelling by car between Minneapolis and Chicago! This suggests that our modelling of Wisconsin's major highways was fairly accurate.

The distance this route corresponds to can be found using our Minneapolis-Chicago and Chicago-Minneapolis entries in *dists_record*.

```
[5]: println(dists_record[:MPS, :CHI], " miles")
     println(dists_record[:CHI, :MPS], " miles")
```

```
447.0 miles
447.0 miles
```

3.2 The Station Locator problem

We create a mixed integer program that allows an EV to, with the help of fast charging stations, successfully traverse each of the 66 OD shortest-path routes generated from the previous MCNF problem. The model optimises the cost of installing charging infrastructure by simulating battery levels in an EV as it drives along each OD route and choosing where charging stations are best located.

Several assumptions and simplifications are made in our model formulation. These include that
- only the shortest-path routes between OD pairs are taken
- EVs always start their journeys with fully charged batteries
  (reasonable, given that Level 1 charging is widely available at home, and EV users would want to set off on long journeys with fully charged batteries)
- a single type of EV, with constant driving range, is used
  (we will later consider the sensitivity of our solutions to EV driving range)
- EV battery depletion and distance travelled have a linear relationship

To promote EV usage in Wisconsin, drivers must be assured of the feasibility of completing long distance journeys across the state. Riding on the fact that the Volkswagen settlement funds will provide the state with the finances for EV infrastructure installation, we assume an unconstrained budget and assign equal weights to the covering of each route.

3.2.1 Station Locator mathematical model

Data
N - set of nodes
R - set of routes
d_{ij} - distance between nodes i and j
c_i - cost of installing a fast charging station at node i
β - maximum battery capacity (in terms of driving range)
M - a large number; set to β
O - route origin node

Variables
X_i - binary indicating if a fast charging station is installed at node i (1 if so; 0 otherwise)
B_i^k - available battery (in terms of driving range) in an EV at node i on OD route k
e_i^k - amount of battery (in terms of driving range) charged to an EV at node i on OD route k

Model

$$\begin{aligned}
\underset{X,B,e}{\text{minimise}} \quad & \sum_{i \in N} c_i X_i & & \\
\text{subject to:} \quad & B_i^k + e_i^k - d_{ij} = B_j^k & \forall i \in N; \forall j \in N; \forall k \in R & (1)\\
& B_i^k + e_i^k \leq \beta & \forall i \in N; \forall k \in R & (2)\\
& e_i^k \leq M X_i & \forall i \in N; \forall k \in R & (3)\\
& B_O^k = \beta & \forall k \in R & (4)\\
& X_i \in \{0, 1\} & \forall i \in N & (5)\\
& B_i^k, e_i^k \geq 0 & \forall i \in N; \forall k \in R & (6)
\end{aligned}$$

The objective of this MIP is to locate a least-cost combination of fast charging stations which ensures that the set of all shortest-path routes between OD pairs, R, is covered. If we assume installation costs to be location-invariant, $c_i = 1$ for all nodes, and the objective function simplifies to become a sum of X_i. Conversely, we can also account for geographical differences in installation costs by assigning different values to c_i for each node. We will hereafter look at both situations in our model, and analyse how their solutions differ. However, we also reiterate that the weighted cost coefficients used are but rough estimates of costs relative to that of installation in Madison (where c_i has been set to be 1).

Constraint (1) describes the battery balance in an EV: the battery level of an EV entering node j is equivalent to its battery level entering node i, plus however much it is charged at i, and minus the distance it has traversed between i and j. Constraint (2) forbids EV batteries from being charged beyond their capacities. Constraint (3) permits battery charging to occur at node i only if a fast charging station is located there. In our model, we set M to be equal to β, which is the upper bound on the EV's battery level. Constraint (4) states that all EVs start their journeys with fully charged batteries. Constraints (5) and (6) are binary and non-negativity variable constraints.

3.2.2 Station Locator implementation

[6]:
```
# ---------- main Mixed Integer Program model ---------- #
using JuMP, Gurobi

function station_locator( = 150, weight = 0)
    ### parameters ###
    #  = EV battery capacity; 150 mi is the default value
    M =


    m2 = Model(Gurobi.Optimizer)
    set_optimizer_attribute(m2, "OutputFlag", 0)

    ### variables ###
    @variable(m2, X[nodes], Bin) # indicates if a fast charging station should␣
↪be installed at a given node

    @variable(m2, B[i in keys(route_record), j in route_record[i][2:end]] >= 0)␣
↪# available battery at node in a route
    @variable(m2, e[i in keys(route_record), j in route_record[i][2:end]] >= 0)␣
↪# power charged to EV at node in a route

    ### constraints ###
    for i in keys(route_record) # for each route
        for j in route_record[i][1] # for each arc in each route
            # battery power balance
            @constraint(m2, B[i, j[1]] + e[i, j[1]] - costs[j] == B[i, j[2]] )
        end
    end

    for i in keys(route_record) # for each route
        for j in route_record[i][2:end] # for each node in each route
            # EV batteries cannot be charged to exceed their capacities
            @constraint(m2, B[i, j] + e[i, j] <=  )

            # charging at a given node can only occur if a fast charging station␣
↪is installed there
            @constraint(m2, e[i, j] <= M*X[j] )
        end

        # at the start of each journey, EV battery is fully charged
        @constraint(m2, B[i, route_record[i][2]] ==  )
    end

    ### objective function ###
    if weight == 0
```

```
        @objective(m2, Min, sum(X)) # apply uniform installation costs
    else
        @objective(m2, Min, sum(weights[i]*X[i] for i in nodes)) # account for␣
 ↪variation in installation costs
    end

    optimize!(m2)

    X_dict = Dict() # initialise empty dictionary to store X values
    for i in nodes
        X_dict[i] = value(X[i])
    end

    return(X_dict, objective_value(m2))
end;
```

We create a helper function, *plotMap*, to visualise the solution of each instance of our Station Locator model. Besides mapping the transport network's nodes and arcs, it also highlights where fast charging stations are sited in blue.

[7]:
```
using CSV, PyCall, PyPlot

# csv with latitude, longitude, adjusted latitude, adjusted longitude in columns␣
 ↪2, 3, 5, 6
raw = CSV.read("Location Data.csv")

latitudes = Dict(zip(nodes, raw[:, 2]) )
longitudes = Dict(zip(nodes, raw[:, 3]) )
plotlat = Dict(zip(nodes, raw[:, 5]) )
plotlon = Dict(zip(nodes, raw[:, 6]) )

base_map = pyimport("mpl_toolkits.basemap")

lightBlue = (150/255, 220/255, 255/255)
lightGreen = (200/255, 255/255, 150/255)

# ---------- Helper Function ---------- #
# generates a map to visualise results of station_locator()
function plotMap(X = 0)
    map = base_map.Basemap(projection = "merc", resolution = "h", llcrnrlat = 41.
 ↪5, llcrnrlon = -93.5,
        urcrnrlat = 46.5, urcrnrlon = -86.5)
    map.drawcoastlines(linewidth = 0.5)
    map.fillcontinents(color = lightGreen, lake_color = lightBlue)
    map.drawstates(linewidth = 0.10)

    # plot highway routes
```

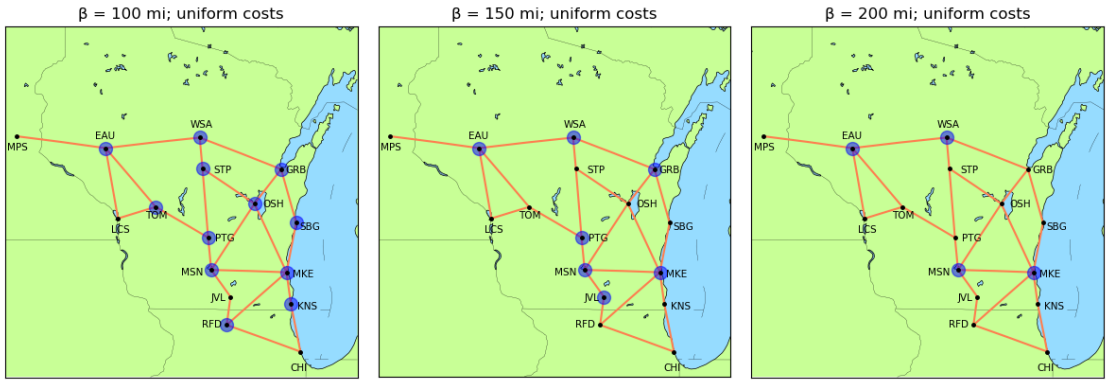
```
    for i in unidir_arcs
        map.drawgreatcircle(longitudes[i[1]], latitudes[i[1]], longitudes[i[2]],␣
↪latitudes[i[2]],
            linewidth = 1.5, color = "coral")
    end

    # plot nodes with fast charging stations
    if X != 0
        for i in nodes
            if X[i] != 0
                map.plot(longitudes[i], latitudes[i], "bo", alpha = 0.5,␣
↪markersize = 10, latlon = true)
            end
        end
    end

    # plot nodes
    for i in nodes
        map.plot(longitudes[i], latitudes[i], "k.", markersize = 5, latlon =␣
↪true)
        plt.text(plotlon[i], plotlat[i], i, fontsize = 7.5)
    end

    return(map)
end;
```

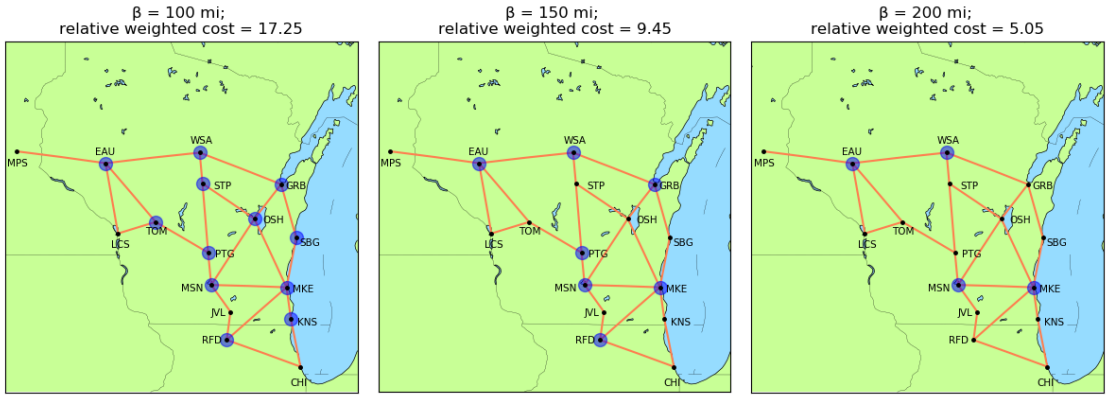
3.2.3 Station Locator results and discussion

We vary our input parameters for each of the six times we run *station_locator* below. The driving range, β, is varied from 100 to 200 miles, in 50-mile increments. We also first assume location-invariant installation costs, before assigning weighted costs to each node in our latter 3 runs. The total weighted cost is given relative to the cost of installing a fast charging station at Madison (which is given a value of 1).

[8]:
```
# assuming uniform installation costs across all nodes
Xdict_100, cost_100 = station_locator(100, 0) # use max battery capacity of 100␣
↪mi
Xdict_150, cost_150 = station_locator(150, 0) # use max battery capacity of 150␣
↪mi
Xdict_200, cost_200 = station_locator(200, 0) # use max battery capacity of 200␣
↪mi

fig = figure(figsize = (12, 12))
subplot(1, 3, 1); plotMap(Xdict_100); title(" = 100 mi; uniform costs")
subplot(1, 3, 2); plotMap(Xdict_150); title(" = 150 mi; uniform costs")
subplot(1, 3, 3); plotMap(Xdict_200); title(" = 200 mi; uniform costs")
tight_layout()
```

```
# assuming installation costs vary by location
Xdict_100, cost_100 = station_locator(100, 1) # use max battery capacity of 100␣
 ↪mi
Xdict_150, cost_150 = station_locator(150, 1) # use max battery capacity of 150␣
 ↪mi
Xdict_200, cost_200 = station_locator(200, 1) # use max battery capacity of 200␣
 ↪mi

fig = figure(figsize = (12, 12))
subplot(1, 3, 1); plotMap(Xdict_100); title(string(" = 100 mi; \nrelative␣
 ↪weighted cost = ", cost_100))
subplot(1, 3, 2); plotMap(Xdict_150); title(string(" = 150 mi; \nrelative␣
 ↪weighted cost = ", cost_150))
subplot(1, 3, 3); plotMap(Xdict_200); title(string(" = 200 mi; \nrelative␣
 ↪weighted cost = ", cost_200))
tight_layout()
```





i. Examining the effect of variations in installation costs

There is limited apparent difference between running the Station Locator model with uniform and weighted costs. The only exceptions are in $\beta = 150$ mi, where a station has been moved from Janesville to Rockford. Thus, the results for the weighted cost scenario do not seem to differ meaningfully from the uniform-cost scenario. Given that fast charger installation costs vary widely our parameters - for instance, the cost of installation at Portage is more than three times that in Chicago, and twice that in Madison - our solutions suggest that the structure of our transport network, and the location of nodes relative to one another, is a much more significant determinant of model results than local variations in installation cost.

ii. Examining the effect of maximum battery capacity/driving range

We discuss the solutions given by the Station Locator model for different β values of both the uniform and weighted-cost scenarios. It appears that, for every increase in EV driving range of 50 miles, our total fast charging station installation cost approximately halves.

The $\beta = 100$ mi instance represents the case where the Wisconsin government is able to roll out fast charging stations quickly, and would like to cater to older EV models which have smaller battery capacities. Of the 16 nodes, only 4 do not have charging stations. Unsurprisingly, Chicago and Minneapolis are 2 of the 4 - as they are our network "bookends", we do not have routes where they are intermediate nodes (i.e. they are always origin or destination nodes). This works for us, though! Such a result is by design of our network, and we don't want to be spending money that belongs to Wisconsin on other states.

The $\beta = 150$ mi instance is representative of the average EV driving range in 2020, as exemplified by the popular 2019 BMW i3 and 2020 Nissan Leaf models. Interestingly, the 'central spine' of the transport network, from Wausau in the north all the way to Rockford in the south, is replete with fast charging nodes. Due to the formulation of our optimisation model, a possible underlying logic explaining this phenomenon is that such a set of placements allows EVs to 'branch off', fully charged, from this main transport artery to reach other corners of the state.

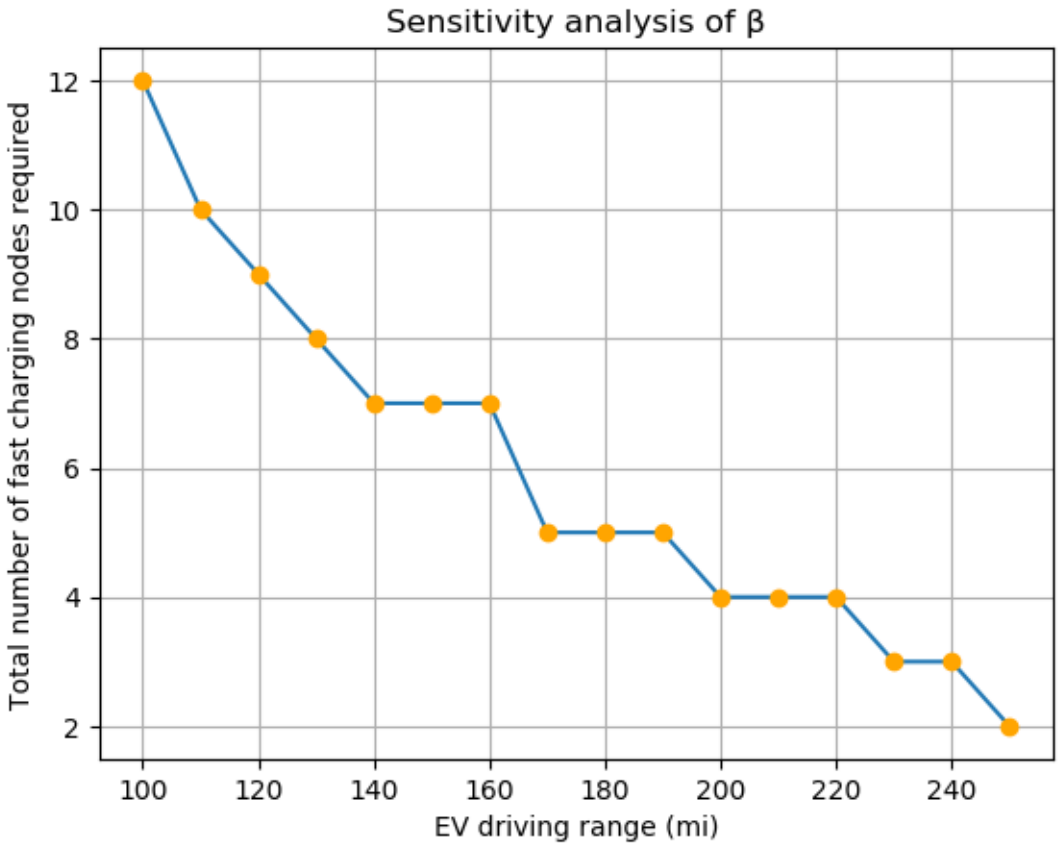
The $\beta = 200$ mi instance simulates the case of a Wisconsin government that is either very forward-looking and confident of EVs' advent, or would like to focus more on the luxury EV market. Madison and Milwaukee continue to qualify as charging station sites even in this optimisation instance. This presents a strong argument for building a robust core of EV fast charging stations in them, as they not only are the two most populated cities in Wisconsin, but also represent critical transit nodes from the perspective of the Wisconsin transportation network.

We perform a sensitivity analysis of β using uniform charging station installation costs, and visualise our results in a plot of number of fast charging stations installed against EV driving range.

[9]:
```
X_store = [] # initialise empty array to store number of charging stations␣
 ↪installed for each  value

for  in 100:10:250
    X_dict, cost = station_locator()

    X_count = 0 # count number of nodes with stations
    for i in nodes
```

```
        if X_dict[i] == 1
            X_count += 1
        end
    end

    push!(X_store, X_count) # add the current X_count value to the X_store array
end
```

[10]:
```
plot(100:10:250, X_store, "-")
plot(100:10:250, X_store, "o", color = "orange")
title("Sensitivity analysis of ")
xlabel("EV driving range (mi)")
ylabel("Total number of fast charging nodes required")
grid()
```



The Station Locator model is especially sensitive to changes in EV driving range between 100 and 140 miles, as observed from the large drop in y-axis values. After which, the model becomes less sensitive: function output values repeatedly plateau for a while before falling again. For each 'step', there is a critical point that the driving range must exceed before it can cover all OD routes with fewer charging stations.

Note that *station_locator* cannot accept β values lower than 99 miles for our instance of the transport network, since this represents the longest edge length (between Eau Claire and Wausau), and a driving range less than that prevents an EV from traversing it even if it is fully charged.

3.3 The Constrained Station Locator problem

Now that the Station Locator solutions have given us an idea of the nodes in which to site fast charging stations, we wish to go a step farther and investigate how we are best able to cover as many shortest-path OD routes as possible with a limited budget. The Constrained Station Locator model defines the budget in terms of how many fast charging stations (for the uniform cost case) or Madison-equivalent fast charging stations (for the weighted cost case) can be installed. Compared to the original SL model, in which coverage of all OD routes is enforced, the CSL model maximises route coverage for a specified budget.

With the exception of the number of fast charging stations which can be installed, CSL takes on the same assumptions and simplifications as those previously discussed for the SL model.

3.3.1 Constrained Station Locator mathematical model

Data
N - set of nodes
R - set of routes
d_{ij} - distance between nodes i and j
c_i - cost of installing a fast charging station at node i
α - budget, in terms of number of fast charging stations that can be installed
β - maximum battery capacity (in terms of driving range)
M - a large number; set to β
m - a small number; set to the negative of the longest OD route's distance
O - route origin node

Variables
X_i - binary indicating if a fast charging station is installed at node i (1 if so; 0 otherwise)
F_k - integer indicating if route k can be traversed successfully by an EV (0 if so; ≥ 1 otherwise)
B_i^k - available battery (in terms of driving range) in an EV at node i on OD route k
e_i^k - amount of battery (in terms of driving range) charged to an EV at node i on OD route k
δ_i^k - binary indicating if B_i^k is negative (1 if so; 0 otherwise)

Model

$$
\begin{aligned}
\underset{X,F,B,e,\delta}{\text{minimise}} \quad & \sum_{k \in R} F_k & & \\
\text{subject to:} \quad & B_i^k + e_i^k - d_{ij} = B_j^k & \forall i \in N; \forall j \in N; \forall k \in R & (1) \\
& B_i^k + e_i^k \leq \beta & \forall i \in N; \forall k \in R & (2) \\
& e_i^k \leq M X_i & \forall i \in N; \forall k \in R & (3) \\
& B_O^k = \beta & \forall k \in R & (4) \\
& \sum_{i \in N} c_i X_i \leq \alpha & & (5) \\
& B_i^k \geq m \delta_i^k & \forall i \in N; \forall k \in R & (6) \\
& F_k \geq \sum_{i \in (N \in k)} \delta_i^k & \forall k \in R & (7) \\
& X_i, \delta_i^k \in \{0, 1\} & \forall i \in N; \forall k \in R & (8) \\
& F_k \in \mathbb{Z} & \forall k \in R & (9) \\
& B_i^k \text{ free} & \forall i \in N; \forall k \in R & (10) \\
& e_i^k \geq 0 & \forall i \in N; \forall k \in R & (11)
\end{aligned}
$$

The objective of this MIP is to minimise the number of shortest-path OD routes that are unfeasible for EVs (i.e. maximise the number of OD routes covered).

Constraints (1) to (4) are identical to those in the Station Locator model. However, the non-negativity constraint for B_i^k has been relaxed such that it is now a free variable. This means that the equality in Constraint (1) and the left-hand side of Constraint (2) can now take on negative values. Constraint (5) imposes a limit on the total cost of fast charging station installation, while (6) and (7) work as a set of logical constraints to enable identification of coverable routes. If B_i^k is negative, signalling that an EV is out of battery and node i along route k cannot actually be reached, Constraint (6) forces the corresponding δ_i^k to equal -1. Constraint (7) then counts how many nodes in a given route are unreachable; if all of them are covered, the objective function obliges F_k to be 0. Otherwise, F_k takes on a positive integer value. Finally, Constraints (8) through (11) describe binary, integer, free, and non-negative variable constraints.

3.3.2 Constrained Station Locator implementation

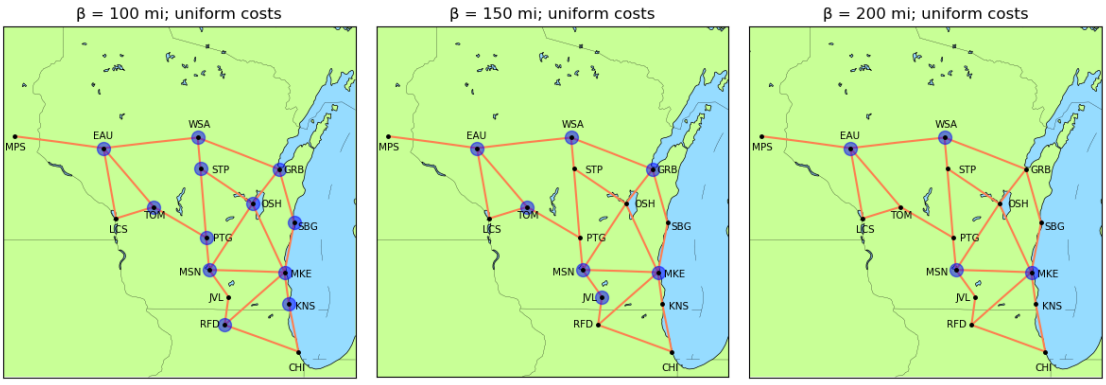
[11]:
```
# ---------- evolved Mixed Integer Program model ---------- #
using JuMP, Gurobi

function constrained_station_locator( = 150, budget = 3, weight = 0)
    ### parameters ###
    #  = EV battery capacity; 150 mi is the default value
    # default budget = 3 uniform-cost charging stations/3 Madison-equivalent␣
 ↪charging stations
    M =
    m = dists_record[:MPS, :CHI] # route with the greatest distance
```

```
   m2 = Model(Gurobi.Optimizer)
   set_optimizer_attribute(m2, "OutputFlag", 0)

   ### variables ###
   @variable(m2, X[nodes], Bin) # indicates if a fast charging station should␣
↪be installed at a given node
   @variable(m2, F[k in keys(route_record)], Int) # if = 0, route k is feasible␣
↪for an EV to traverse fully

   @variable(m2, B[i in keys(route_record), j in route_record[i][2:end]] ) #␣
↪available battery at node in a route
   @variable(m2, e[i in keys(route_record), j in route_record[i][2:end]] >= 0)␣
↪# power charged to EV at node in a route
   @variable(m2, [i in keys(route_record), j in route_record[i][2:end]], Bin) #␣
↪indicates if B[i, j] is positive

   ### constraints ###
   for i in keys(route_record) # for each route
       for j in route_record[i][1] # for each arc in each route
           # battery power balance
           @constraint(m2, B[i, j[1]] + e[i, j[1]] - costs[j] == B[i, j[2]] )
       end
   end

   for i in keys(route_record) # for each route
       for j in route_record[i][2:end] # for each node in each route
           # EV batteries cannot be charged to exceed their capacities
           @constraint(m2, B[i, j] + e[i, j] <=  )

           # charging at a given node can only occur if a fast charging station␣
↪is installed there
           @constraint(m2, e[i, j] <= M*X[j] )

           @constraint(m2, B[i, j] >= -m*[i, j])
       end
       # if battery levels are positive at each stage of route i, allow F[i] to␣
↪= 0
       @constraint(m2, sum([i, j] for j in route_record[i][2:end]) <= F[i])

       # at the start of each journey, EV battery is fully charged
       @constraint(m2, B[i, route_record[i][2]] ==  )
   end

   # budget constraint - choice between uniform or weighted costs
```

```
    if weight == 0
        @constraint(m2, budget_constr, sum(X) <= budget )
    else
        @constraint(m2, budget_constr, sum(weights[i]*X[i] for i in nodes) <=␣
↪budget )
    end

    ### objective function ###
    @objective(m2, Min, sum(F))

    optimize!(m2)

    X_dict = Dict() # initialise empty dictionary to store X values
    for i in nodes
        X_dict[i] = value(X[i])
    end

    F_dict = Dict() # initialise empty dictionary to store F values
    for k in keys(route_record)
        F_dict[k] = value(F[k])
    end

return(X_dict, F_dict)
end;
```

We would like to compare the CSL model's solutions against those of the SL model. This is done by adjusting the budget parameter value in *constrained_station_locator* to correspond to the number of charging stations the SL model previously determined was needed - in the case of uniform installation costs, we needed 12, 7, and 4 stations for driving ranges of 100, 150, and 200 miles respectively.

[12]:
```
# assuming uniform installation costs across all nodes
Xdict_100, Fdict_100 = constrained_station_locator(100, 12, 0) # use max battery␣
↪capacity of 100 mi
Xdict_150, Fdict_150 = constrained_station_locator(150, 7, 0) # use max battery␣
↪capacity of 150 mi
Xdict_200, Fdict_200 = constrained_station_locator(200, 4, 0) # use max battery␣
↪capacity of 200 mi

fig = figure(figsize = (12, 12))
subplot(1, 3, 1); plotMap(Xdict_100); title(" = 100 mi; uniform costs")
subplot(1, 3, 2); plotMap(Xdict_150); title(" = 150 mi; uniform costs")
subplot(1, 3, 3); plotMap(Xdict_200); title(" = 200 mi; uniform costs")
tight_layout()
```



Comparing our new solutions with those generated from the SL model, we observe that a charging station at Portage has moved to Tomah in the $\beta = 150$mi instance. The SL and CSL solutions are otherwise identical.

Let's check the *F_dict* values of the $\beta = 150$mi instance of the CSL to confirm that all OD routes were covered with our new model. We create a function, *count_covered_routes*, that checks how many F values equal 0 (i.e. how many routes are fully covered). If all of our OD routes are covered, *count_covered_routes* will return the maximum value of 66.

[13]:
```
# ---------- Helper Function ---------- #
# counts the number of OD routes a solution for a given instance of␣
 ↪constrained_station_locator covers
function count_covered_routes(F_dict)
    count = 0

    for i in keys(route_record)
        if F_dict[i] == 0
            count += 1
        end
    end
    return(count)
end

println("OD routes covered for  = 150mi instance: ",␣
 ↪count_covered_routes(Fdict_150))
```

```
OD routes covered for  = 150mi instance: 66
```

So the $\beta = 150$mi instance indeed covered all 66 OD routes. Since both the SL and CSL models have returned different sets of charging station locations that can nonetheless cover all routes, we conclude that the Wisconsin transport network built in this project is capable of returning multiple optimal solutions for a given instance.

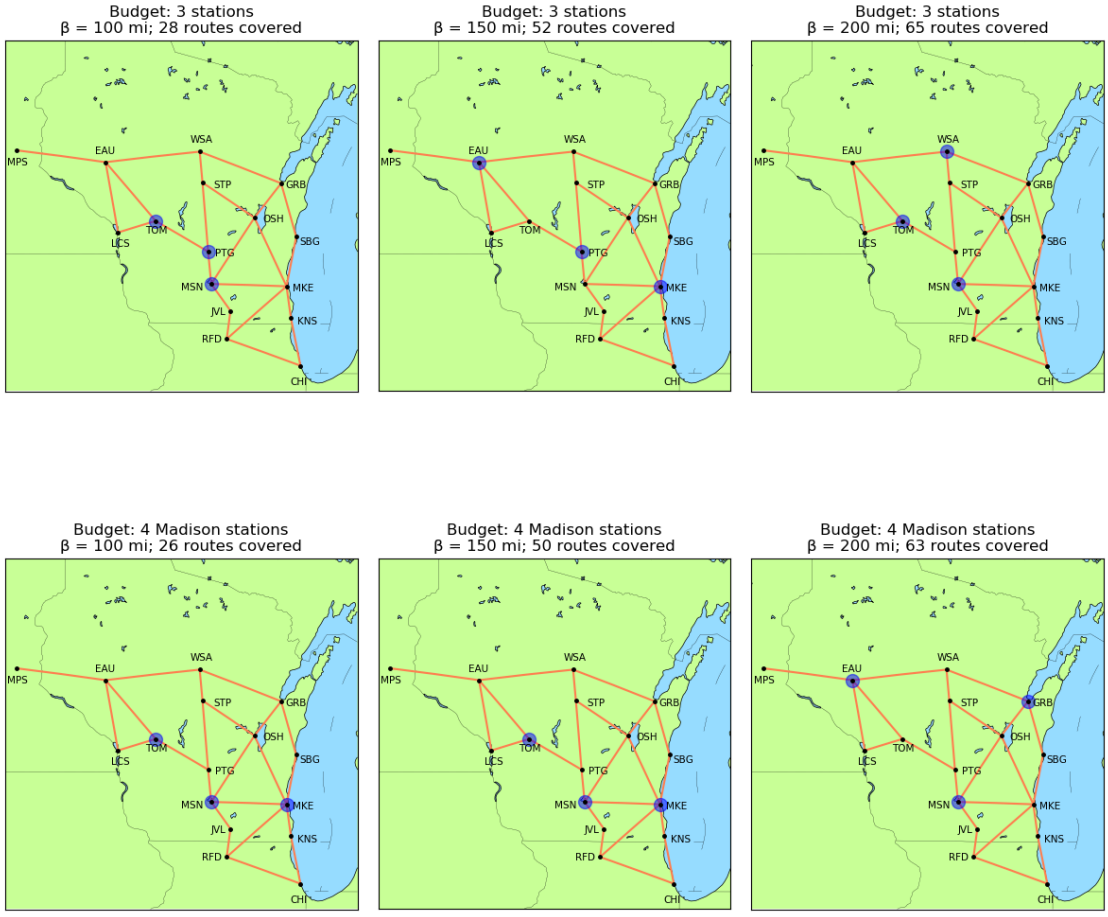
3.3.3 Constrained Station Locator results and discussion

With the validity of the CSL model now verified, we turn to investigate how EV driving range affects the optimal siting of a limited number of charging stations. Given that, for $\beta = 200$mi, only 4 stations are needed for total OD route coverage, we limit our budget to 3 charging stations when using uniform installation costs.

When choosing to use weighted installation costs instead, we 'inflate' our budget to the equivalent of installing 4 Madison fast charging stations. This is because running *station_locator* with $\beta = 200$mi gave a relative total weighted cost of 5.05 earlier, and we want to account for how the average cost of installing a fast charging station is now greater than 1.

[14]:
```
# assuming uniform installation costs across all nodes; budget for 3 charging␣
 ↪stations
Xdict_100, Fdict_100 = constrained_station_locator(100, 3, 0) # use max battery␣
 ↪capacity of 100 mi
Xdict_150, Fdict_150 = constrained_station_locator(150, 3, 0) # use max battery␣
 ↪capacity of 150 mi
Xdict_200, Fdict_200 = constrained_station_locator(200, 3, 0) # use max battery␣
 ↪capacity of 200 mi

fig = figure(figsize = (12, 12))
subplot(1, 3, 1); plotMap(Xdict_100); title(string("Budget: 3 stations\n = 100␣
 ↪mi; ",
                                                    ␣
 ↪count_covered_routes(Fdict_100), " routes covered"))
subplot(1, 3, 2); plotMap(Xdict_150); title(string("Budget: 3 stations\n = 150␣
 ↪mi; ",
                                                    ␣
 ↪count_covered_routes(Fdict_150), " routes covered"))
subplot(1, 3, 3); plotMap(Xdict_200); title(string("Budget: 3 stations\n = 200␣
 ↪mi; ",
                                                    ␣
 ↪count_covered_routes(Fdict_200), " routes covered"))
tight_layout()


# assuming installation costs vary by location; budget for 4 Madison-equivalent␣
 ↪stations
Xdict_100, Fdict_100 = constrained_station_locator(100, 4, 1) # use max battery␣
 ↪capacity of 100 mi
Xdict_150, Fdict_150 = constrained_station_locator(150, 4, 1) # use max battery␣
 ↪capacity of 150 mi
Xdict_200, Fdict_200 = constrained_station_locator(200, 4, 1) # use max battery␣
 ↪capacity of 200 mi

fig = figure(figsize = (12, 12))
```

```
subplot(1, 3, 1); plotMap(Xdict_100); title(string("Budget: 4 Madison stations\n␣
 ↪= 100 mi; ",
                                                                    ␣
 ↪count_covered_routes(Fdict_100), " routes covered"))
subplot(1, 3, 2); plotMap(Xdict_150); title(string("Budget: 4 Madison stations\n␣
 ↪= 150 mi; ",
                                                                    ␣
 ↪count_covered_routes(Fdict_150), " routes covered"))
subplot(1, 3, 3); plotMap(Xdict_200); title(string("Budget: 4 Madison stations\n␣
 ↪= 200 mi; ",
                                                                    ␣
 ↪count_covered_routes(Fdict_200), " routes covered"))
tight_layout()
```



i. Examining the effect of variations in installation costs

Even with an inflated budget of 4 Madison-equivalent charging stations, the variable installation cost instances were unable to afford the fast charging stations recommended by their correspond-

ing uniform cost instances. For example, the $\beta = 100$ mi, uniform cost instance suggests building charging stations at Madison, Portage, and Tomah, but this would incur a total relative cost of 5.0 in the weighted cost instance. Instead, the $\beta = 100$ mi, weighted cost instance came up with a Madison, Milwaukee, and Tomah combination with a total relative cost of 3.85 Madison stations. Since the CSL model's weighted cost instances have a more constrained budget than their uniform cost counterparts (because the average relative cost of a charging station is greater than 1 when weighted), it is reasonable that they cover fewer OD routes.

One possible follow-up to this project is to re-evaluate the cost coefficients, and scale them down to have an average value of 1. This will enable direct, charging station-to-charging station comparison between the uniform and weighted cost instances.

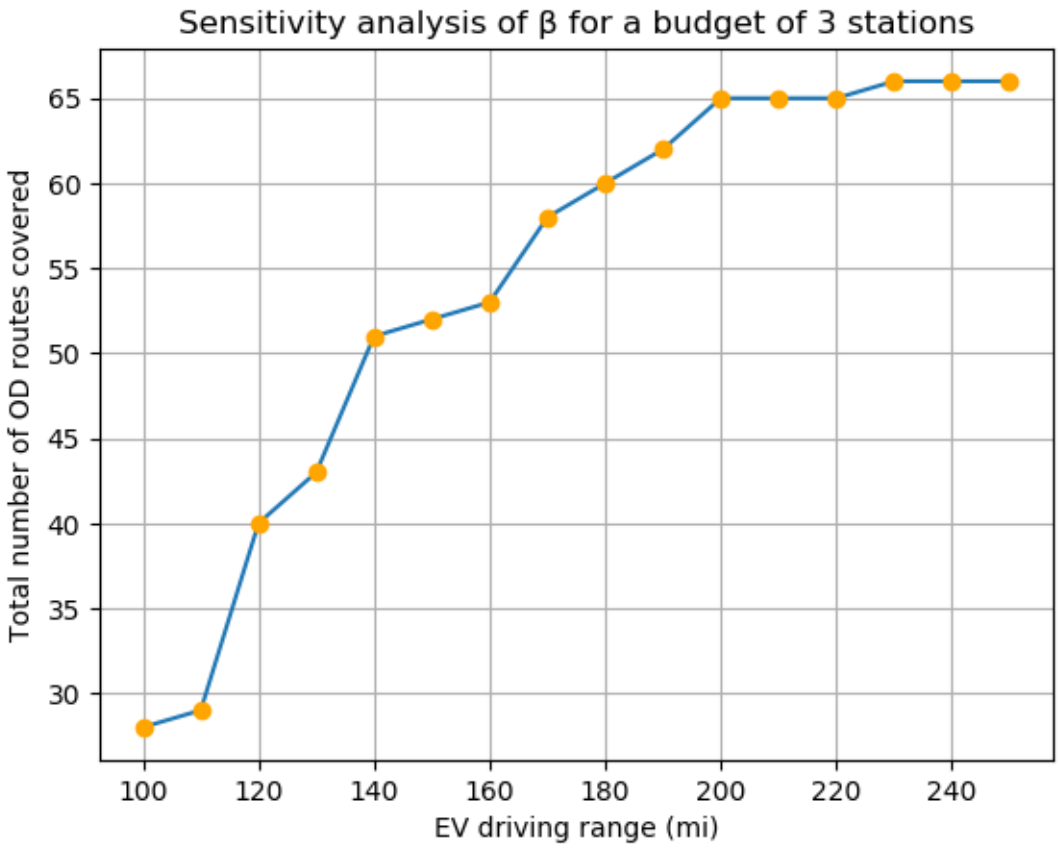
ii. Examining the effect of maximum battery capacity/driving range

The exact mechanisms by which changes to β affect the CSL model's solutions is less clear. However, there does seem to be an overall trend where larger battery capacities encourage charging station installation along the northeastern bend of the Wisconsin transport network, from Wausau down to Milwaukee.

By setting our budget to a constant of 3 uniform-cost fast charging stations, we perform a sensitivity analysis of β, and visualise our results in a plot of number of OD routes covered against EV driving range.

[15]:
```
F_store = [] # initialise empty array to store number of OD routes covered for␣
 ↪each  value

for  in 100:10:250
    X_dict, F_dict = constrained_station_locator(, 3, 0) # budget of 3 charging␣
 ↪stations; uniform installation costs

    push!(F_store, count_covered_routes(F_dict))
end
```

[16]:
```
plot(100:10:250, F_store, "-")
plot(100:10:250, F_store, "o", color = "orange")
title("Sensitivity analysis of  for a budget of 3 stations")
xlabel("EV driving range (mi)")
ylabel("Total number of OD routes covered")
grid()
```



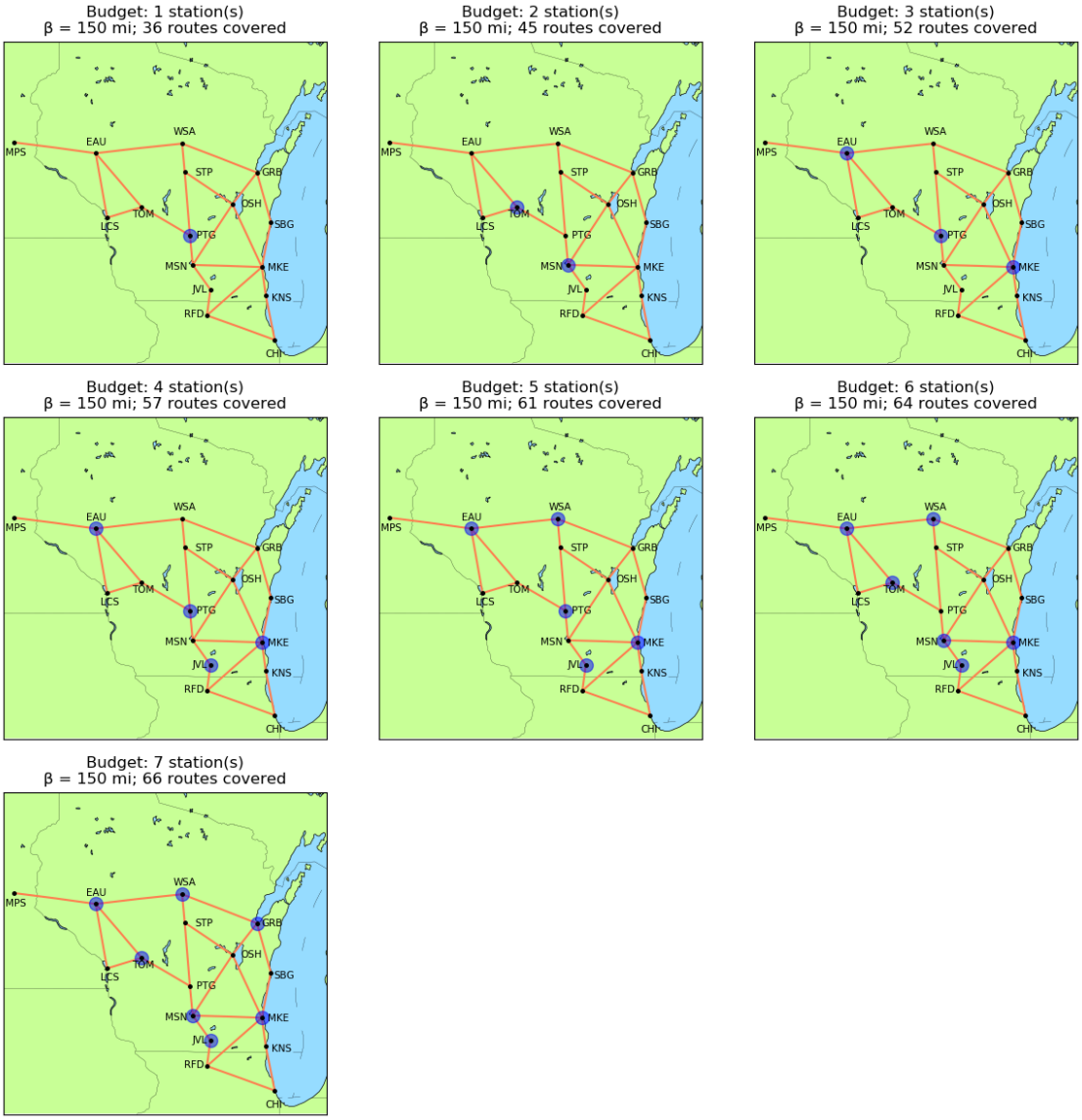
Similar to our sensitivity analysis of β in the SL model, we note steep changes in the y-axis values between an EV driving range of 100 to 140 miles. For EV manufacturers, this may be useful information: assuming the Wisconsin network used in this project can be generalised to statewide transportation networks, our observation suggests that increasing battery capacities to cover up to 140 miles can significantly improve the number of feasible routes available to EV users in the US.

iii. Examining the effect of installation budget

Finally, we explore how changes in the budget, α, affect our solution to the CSL problem. For a fixed β of 150 miles, we iteratively increment the budget by 1 fast charging station and run our model. This can be useful if, say, the Wisconsin government does intend to eventually cover all routes for an assumed EV driving range of 150 miles, but wants to roll out charging infrastructure construction in stages.

[17]:
```
fig = figure(figsize = (12, 12))
for i in 1:7
    Xdict_150, Fdict_150 = constrained_station_locator(150, i, 0)
    subplot(3, 3, i); plotMap(Xdict_150); title(string("Budget: ", i, "␣
 ↪station(s)\n = 150 mi; ",
```

```
 ↪count_covered_routes(Fdict_150), " routes covered"))
end
tight_layout()
```



From our plotted networks, we can plausibly implement a 3-stage installation plan where fast charging stations are initially built at Eau Claire, Milwaukee, and Portage. This is followed by the addition of Janesville and Wausau stations in the intermediate stage, followed finally by Green Bay and Madison.

4 Conclusion

This report looked into how EV fast charging stations should be optimally sited within the Wisconsin transport network to facilitate long distance trips. By reproducing the logic of an EV battery, the mathematical models introduced were able to identify critical nodes for which locating EV charging infrastructure would be highly desirable, while operating with both unconstrained and constrained installation budgets.

Our findings suggest that the choice of EV driving range value used in locating charging stations has dramatic influence over the solutions generated and their resulting implications. In addition, while regional variations in cost do affect model results, the actual structure of the network being tested seems to be of even greater importance. All of these have implications for EV-related policies. With regards to Wisconsin specifically, Madison and Milwaukee emerged as key transit points through which many long distance routes become feasible. To create connectivity in the western and northern parts of the state, however, construction of EV charging stations in Portage/Tomah and Wausau, respectively, is required, though costly.

Future Directions

One follow-up which should be pursued is to investigate how to ascertain, and then find, the existence of other optimal solutions for a given instance of our MIPs. This is because, while different sets of nodes may technically return the same objective value in the SL and CSL problems, on-the-ground realities may have us prefer one solution over another.

A fruitful next step will involve testing out our models on transport networks in other states, before scaling up the network size to cover larger regions such as the Midwest. However, given that the addition of even a single node creates exponentially more variables in the SL and CSL models, we are presently unsure of how much computational complexity this will involve. It is possible that very large networks will result in impractical run times for our programs. Besides which, it would be valuable to include travel demand considerations in our model, possibly by assigning weights to the flow on each edge. Alternatively, we can also relax our shortest-path approach to the problem, and allow EVs to take longer routes between OD pairs as long as they do not run out of battery. An ambitious, fully-fledged model will incorporate Level 2 AC chargers into the equation, not to mention address concerns about the impact and interfacing of EV charging infrastructure with the electrical grid. Such a model will be better able to tackle the complex, multi-faceted issue of how to approach policymaking in EV charging infrastructure.

5 References

[1] *U.S. Light-Duty Advanced Technology Vehicle (ATV) Sales (2011–2019)*, Advanced Technology Vehicle Sales Dashboard, Alliance of Automobile Manufacturers, Aug. 20, 2019. [Online]. Available: https://autoalliance.org/energy-environment/advanced-technology-vehicle-sales-dashboard/

[2] *Alternative Fueling Station Locator*, Alternative Fuels Data Center, U.S. DOE's Office of Energy Efficiency and Renewable Energy. [Online]. Available: https://afdc.energy.gov/stations/#/corridors

[3] P. Ducharme and C. Kargas, "Feasibility of a Pan-Canadian Network of DC Fast Charging Stations for EVs," in *World Electric Vehicle Journal Vol. 8*. EVS29 Symposium, June 19-22, 2016, Montréal, Canada.

[4] WisDOT Office of Public Affairs, *State requests public comment on electric vehicle charging station grant program*, State of Wisconsin Department of Transportation, Jan. 27, 2020. [Online]. Available: https://wisconsindot.gov/Pages/about-wisdot/newsroom/news-rel/012720-VWEVC.aspx.

[5] M. Nicholas, "Estimating electric vehicle charging infrastructure costs across major U.S. metropolitan areas," International Council on Clean Transportation, Publication. Working Paper 2019-14, Aug. 2019.

[6] M. Smith and J. Castellano, "Costs Associated With Non-Residential Electric Vehicle Supply Equipment," New West Technologies, LLC, Report. DOE/EE-1289, Nov. 2015.