524 Final Project

May 8, 2018

Selecting Optimal Data Center within Continental US
ISyE/ECE/CS -- Introduction to Optimization -- Spring 2018

1 Selecting Optimal Data Center within Continental US

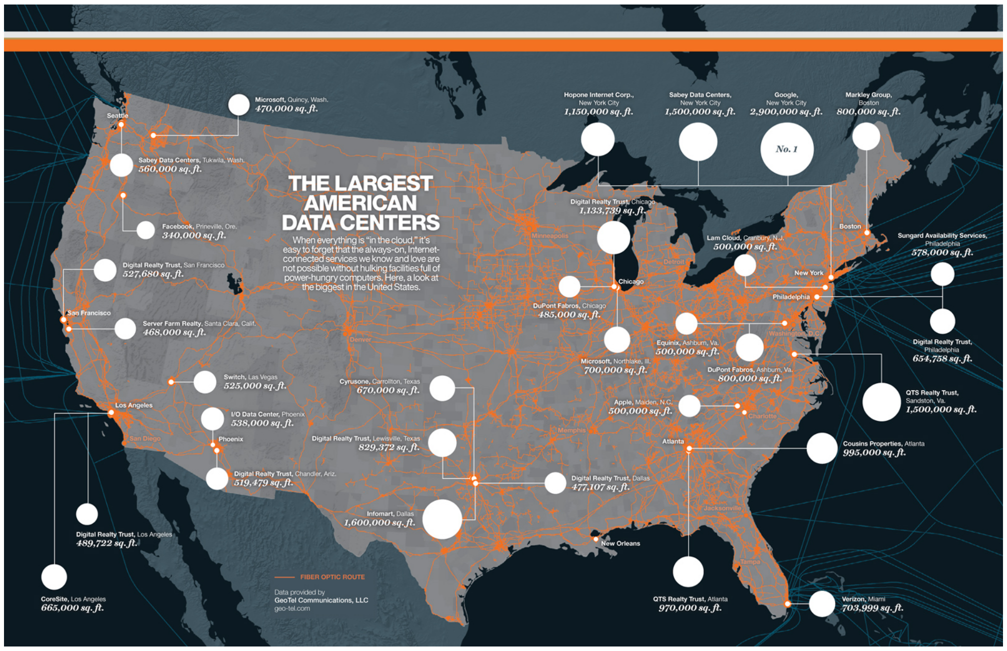
Team: * Yuliia Kapeliushna (kapeliushuna@wisc.edu) * Thomas Hansen (thansen8@wisc.edu) * Julian Nazareth (jnazareth@wisc.edu) * Zuf Wang (xwang523@wisc.edu)

1. Yuliia Kapeliushna (kapeliushuna@wisc.edu)
2. Thomas Hansen (thansen8@wisc.edu)
3. Julian Nazareth (jnazareth@wisc.edu)
4. Zuf Wang (xwang523@wisc.edu)

1.0.1 Table of Contents

2 Introduction

Cloud computing represents an attractive solution for many organizations since it means they no longer have to run and manage their own data centers. Rather, they can lease server space on


THE LARGEST AMERICAN DATA CENTERS
When everything is "in the cloud," it's easy to forget that the always-on, Internet-connected services we know and love are not possible without hulking facilities full of power-hungry computers. Here, a look at the biggest in the United States.
Microsoft, Quincy, Wash. 470,000 sq. ft.
Seattle
Sabey Data Centers, Tukwila, Wash. 560,000 sq. ft.
Facebook, Prineville, Ore. 340,000 sq. ft.
Digital Realty Trust, San Francisco 527,680 sq. ft.
San Francisco
Server Farm Realty, Santa Clara, Calif. 468,000 sq. ft.
Switch, Las Vegas 525,000 sq. ft.
Los Angeles
San Diego
I/O Data Center, Phoenix 538,000 sq. ft.
Phoenix
Digital Realty Trust, Chandler, Ariz. 519,479 sq. ft.
Digital Realty Trust, Los Angeles 489,722 sq. ft.
CoreSite, Los Angeles 665,000 sq. ft.
Denver
Cyrusone, Carrollton, Texas 670,000 sq. ft.
Digital Realty Trust, Lewisville, Texas 829,372 sq. ft.
Infomart, Dallas 1,600,000 sq. ft.
FIBER OPTIC ROUTE
Data provided by GeoTel Communications, LLC geo-tel.com
Hopone Internet Corp., New York City 1,150,000 sq. ft.
Sabey Data Centers, New York City 1,500,000 sq. ft.
Google, New York City 2,900,000 sq. ft.
No. 1
Markley Group, Boston 800,000 sq. ft.
Digital Realty Trust, Chicago 1,133,739 sq. ft.
Minneapolis
Detroit
Boston
Lam Cloud, Cranbury, N.J. 500,000 sq. ft.
Sungard Availability Services, Philadelphia 578,000 sq. ft.
New York
Chicago
Philadelphia
DuPont Fabros, Chicago 485,000 sq. ft.
Washington, D.C.
Equinix, Ashburn, Va. 500,000 sq. ft.
Microsoft, Northlake, Ill. 700,000 sq. ft.
DuPont Fabros, Ashburn, Va. 800,000 sq. ft.
Digital Realty Trust, Philadelphia 654,758 sq. ft.
Apple, Maiden, N.C. 500,000 sq. ft.
Charlotte
QTS Realty Trust, Sandston, Va. 1,500,000 sq. ft.
Memphis
Atlanta
Cousins Properties, Atlanta 995,000 sq. ft.
Digital Realty Trust, Dallas 477,107 sq. ft.
Jacksonville
New Orleans
Tampa
QTS Realty Trust, Atlanta 970,000 sq. ft.
Verizon, Miami 703,999 sq. ft.

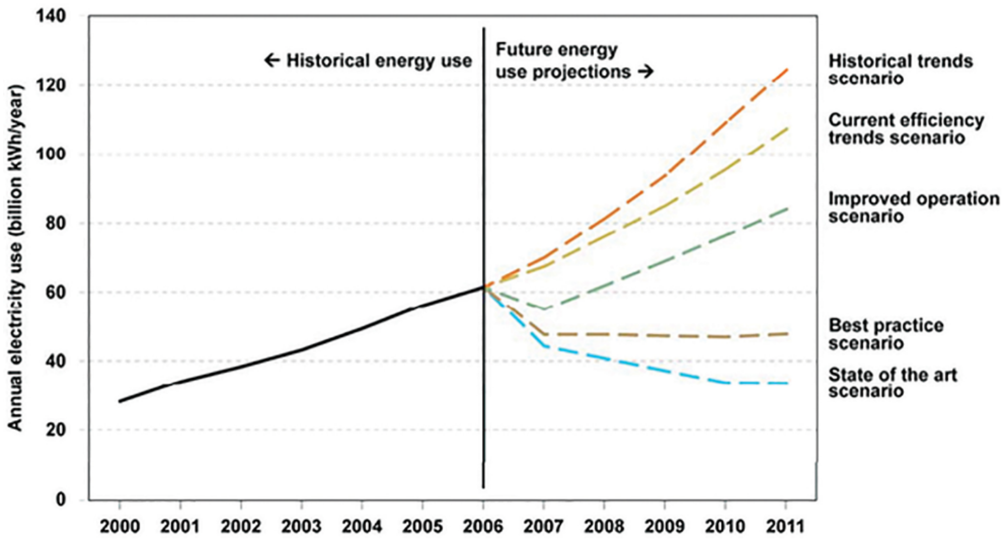
alt text

cloud computing providers or they can build their server space out to incorporate the structure of a content distributions network (CDN) or build caches for data intensive services. Networks like these can be more practical and lead to lower latencies than running completely in-house data centers or running everything from one location.

A data center is a collection of servers that stores, manages, and distributes its content both to the organization and to its consumers. Major service providers such as Amazon (Citation), Google (Citation), and Facebook (Citation) all have numerous data centers to serve their consumer, lower latency, and ensure backup data centers are available in case one fails. Additionally services like Netflix have many smaller "caches" that can greatly reduce congestion and latency by distributing many of these across the country and sharing large video files from them (Citation). Since data centers are the backbone of these companies, making data centers run as effectively as possible is an important goal to achieve.

To run these data centers effectively, we must consider both costs of operation (such as cooling servers and construction), as well as latency to access the user (approximated by distance). Although cooling costs may be very low in the Arctic Circle (Citation), it is important to have data centers where there is the demand to reduce latency and overall network congestion. Since it is best to locate data centers in cooler climate to reduce cooling cost, we expect to see our results will yield data centers spread evenly across the US but with a northern trend. However, in reality, constraints beyond cooling costs, such as taxation, demand, and service type will play a role in data center location determination. See image 1 below for a map of a few selected US-based data centers (Citation).

We carry out this project from the perspective of an internet service provider (similar to the AWS service) with a goal of selecting the optimal data center locations to serve a series of cities with set amount of population. In particular, we aim to minimize overall cost while maximizing download speed. Overall cost is split into the fixed cost of doing business and variable cost is simplified as cooling cost. The simplication of variable cost is because the majority of energy consumption goes towards cooling the server sets. See image below to get an idea of electricty consumption graph.bb



alt text

consumption at data centers (Citation). Our approach seeks to understand the impact of the cost of operations, fixed cost of having a data center, and service distance in data center location determination.

We frame our project as a set covering mixed integer program. Numerous assumptions are made so as to keep our project within a workable scope. We will model this problem progressively with increasing modeling complexity and will showcase each step in the process.

# 2. Data Our dataset contains top 50 most populous US metropolitan areas.

2.0.1 Real Data

- **Metropolitan Population:** Taken from US Census.
- **City Long-Lat:** Taken from SimpleMaps, which offers longitude and latitude data for most major US cities.

2.0.2 Synthetic Data

These are data that we could not acquire without substantial upfront effort to standardize and preprocess. Because this is a modeling focused project, we elected to simplify the data preprocessing task by making up our own cost data. Real cost may differ drastically from ours.

- **Fixed Cost:** We used `randbetween()` in Excel to produce fixed costs.
- **Variable Cost (Cooling Cost):** We used the same `randbetween()` in Excel to come up with baseline variable cost. Because we assumed that cooling cost was the only component of variable cost, we scaled the random number generator to a city's latitude like so `randbetween() * latitude`. This is because the further south the data centers are located, the more it costs to cool data centers because of warmer climate.

# 3. Mathematical Model

This section will provide an overview of the mathematical models we used to optimize the required server power needed for our optimization problem, as well as it will enumerate the different sets of variables as we introduce them into our models.

3A. Assumptions In order to produce a functional model within the limited time, we made numerous simplifying assumptions.

1. Every city is within 800 kilometers of a data center (for Models 0 and 1 only).

2. For Models 2 and 3, the shorter the distance between the city and data center the better.
3. Each person on average uses 0.001 TB of storage.
4. In reality, data centers do not have to be located in the center of the city and could be a few miles out. For the sake of hiring talent and the scope of this project, we assume that data centers are within a metro area.
5. Each data center has a maximum storage capacity of 10^6 TB.
6. Real world cloud storage work very differently then our assumption, which functions closer to physical storage warehouses.
7. Fixed costs are greater than variable costs.
8. Demand for storage is population dependent. We make no distinction between individual user and group user. This in reality is not true since some cities will have higher demand due to local industry and cultural trends.

3B. Model

2.0.3 Global Parameters and Constants

- m denotes total number of cities.
- $consumed = 0.001$
- $distance\_limit = 800$
- $r = 6372.8$. Earth radius.

2.0.4 Calculating Distance

We elected to calculate distance using this Haversine Formula in order to incorporate earth's curvature.

$$dist_{i,j} = 2 * r * \sin^{-1}\sqrt{\sin((lat_j - lat_i)/2)^2 + \cos(lat_i) + \cos(lat_j) + \sin((lon_j - lon_i)/2)^2} \quad \forall i, j \in \{1...m\}$$

where: * lat and lon represents latitude and longitude of each city, respectively. * $dist_{i,j}$ represents the matrix of distances between city i and j.

Given the assumption that each data center must be located within 800 kilometers of the cities it serves, we use $set_{i,j}$, a binary 2d array, to denote set of cities i and j that are less than 800 kilometers from each other. If the distance less than 800 kilometers, the set is 1, otherwise is 0.

Model 0.0 This model seeks to find the minimal number of data centers needed based on the distance limit.

Decision Variable We use c_i as binary variable to represent which city i will host a data center.

$$c_i \in \{0, 1\} \quad \forall i \in \{1...m\}$$

Constraints This contraint ensures that each city is within 800 kilometers of a data center. It does not need to be served by the closest data center.

$$set_{i,j} * c_i \geq 1 \quad \forall i,j \in \{1...m\}$$

Objective We seek to minimize the number of data centers needed based purely on distance.

$$f(c) = \sum_{1}^{i=m} c_i \quad \forall i \in \{1...m\}$$

Standard Form

$$\begin{array}{lll} \text{minimize} & \sum_{1}^{i=m} c_i & \\ \text{subject to:} & \text{set} * c_i \geq 1 & \forall i,j \in \{1...m\} \end{array}$$

Model 1.0 This model seeks to allocate storage to each data center based on population census without consideration of fixed cost. It is built upon Model 0.

Matrices from Model 0.0 $selected_{i,j}$ $\forall i,j \in \{1...m\}$ stores the result of c decision variable.

$selset_{i,j}$ $\forall i,j \in \{1...m\}$ stores the optimal layout that gurantees the minimal data centers in US that serves all cities within 800 kilometers.

Decision Variables We use $stor_i$ to denote storage size allocated to each data center.

$$stor_i \geq 0 \quad \forall i \in \{1...m\}$$

Constraints Storage upper bound constraint, ensuring that cities that do not host data centers do not store data.

$$stor_i \leq selected_{i,j} * max \quad \forall i,j \in \{1...m\}$$

Storage lower bound constraint to meet minimal demand.

$$consumed * selset_{i,j} * census_j \leq stor_i \quad \forall j \in \{1...m\}$$

Objective We use $cooling_i$ to denote cooling cost of hosting a data center in city i. We seek to minimize overall cooling cost.

$$f(c) = \sum_{1}^{i=m} stor_i * cooling_i$$

Standard Form

$$\begin{array}{lll} \text{minimize} & \sum_{1}^{i=m} stor_i * cooling_i & \\ \text{subject to:} & stor_i \leq selected_{i,j} * max & \forall i,j \in \{1...m\} \\ & consumed * selset_{i,j} * census_j \leq stor_i & \forall j \in \{1...m\} \end{array}$$

Model 2.0 Finding data center with consideration of fixed cost alone along with various service constraints that each data center has to fulfill.

Weight factor $\lambda = 1$ to place equal weight for fixed cost and distance.

Decision Variables We use c_i as binary variable to represent which city i will host a data center.

$$c_i \in \{0,1\} \quad \forall i \in \{1...m\}$$

We use $dc_{i,j}$ to denote set of data centers i that serves cities j.

$$dc_{i,j} \in \{0,1\} \quad \forall i \in \{1...m\}$$

Constraints This contraint ensures that each city is within 800 kilometers of a data center. It does not need to be served by the closest data center.

$$set_{i,j} * c_i \geq 1 \quad \forall i,j \in \{1...m\}$$

For the sake of not overburdening any single data center, we elect to have each data center serve no more than half of all client cities.

$$\sum_{1}^{i=m} dc_{i,j} \leq c_i * \frac{m}{2} \quad \forall j \in \{1...m\}$$

This constraint ensures that each city is serviced.

$$\sum_{1}^{j=m} dc_{i,j} = 1 \quad \forall i \in \{1...m\}$$

Ensures each city that hosts a data center serves itself. In other words, we are matching the diagonal values.

$$dc_{i,i} = c_i \quad \forall i \in \{1...m\}$$

Objective Here we minimize both variable cost and distance from data center i to city j. We wished to more accurately simulate real world fixed cost, hence the 10^6 multiplier.

$$f(c) = \lambda * \sum_{1}^{i=m}(fixed_i * max * c_i) + \sum_{1}^{i=m}\sum_{1}^{j=m}(dist_{i,j} * dc_{i,j})$$

Standard Form Combined, our model yields:

$$\begin{aligned}
\text{minimize} \quad & \lambda * \sum_{1}^{i=m}(fixed_i * max * c_i) + \sum_{1}^{i=m}\sum_{1}^{j=m}(dist_{i,j} * dc_{i,j}) & \\
\text{subject to:} \quad & dc_{i,i} = c_i & \forall i \in \{1...m\} \\
& \sum_{1}^{j=m} dc_{i,j} = 1 & \forall i \in \{1...m\} \\
& \sum_{1}^{i=m} dc_{i,j} \leq c_i * \frac{m}{2} & \forall j \in \{1...m\} \\
& set_{i,j} * c_i \geq 1 & \forall i,j \in \{1...m\}
\end{aligned}$$

Model 3.0 This is our most complex model which combines the previous constraints and decision variables.

Weight factor $\lambda = 1$ to place equal weight for fixed cost and distance.

Decision Variables We use c_i as binary variable to represent which city i will host a data center.

$$c_i \in \{0, 1\} \quad \forall i \in \{1...m\}$$

We use $dc_{i,j}$ as binary variable to represent data center in city i serving city j.

$$dc_{i,j} \in \{0, 1\} \quad \forall i, j \in \{1...m\}$$

We use $stor_i$ to denote storage size allocated to each data center.

$$stor_i \geq 0 \quad \forall i \in \{1...m\}$$

Constraints This contraint ensures that each city is within 800 kilometers of a data center. It does not need to be served by the closest data center.

$$set_{i,j} * c_i \geq 1 \quad \forall i, j \in \{1...m\}$$

For the sake of not overburdening any single data center, we elect to have each data center serve no more than half of all client cities.

$$\sum_{1}^{i=m} dc_{i,j} \leq c_i * \frac{m}{2} \quad \forall j \in \{1...m\}$$

This constraint ensures that each city is serviced.

$$\sum_{1}^{j=m} dc_{i,j} = 1 \quad \forall i \in \{1...m\}$$

This constraints requires each data center to be allocated a minimal amount of storage that would meet the demand of cities served. Demand is represented by population.

$$consumed * \sum_{1}^{j=m} dc_{i,j} * census_j \leq stor_i \quad \forall i \in \{1...m\}$$

Ensures that each city that hosts a data center serves itself. In other words, we are matching the diagonal values.

$$dc_{i,i} = c_i \quad \forall i \in \{1...m\}$$

Objective Minimize all factors: variable cost, fixed cost, and distance from data center i to serviced city j.

$$f(c) = \lambda * \sum_{1}^{i=m} \sum_{1}^{j=m} (dist_{i,j} * dc_{i,j}) + \sum_{1}^{i=m} (cooling_i * stor_i + fcost_i * c_i)$$

Standard Form Combined, our model yields:

$$
\begin{aligned}
\text{minimize} \quad & \lambda * \sum_{1}^{i=m} \sum_{1}^{j=m} dist_{i,j} * dc_{i,j} + \sum_{1}^{i=m} (cooling_i * stor_i + fcost_i * c_i) & \\
\text{subject to:} \quad & set_{i,j} * c_i \geq 1 & \forall i,j \in \{1...m\} \\
& \sum_{1}^{i=m} dc_{i,j} \leq c_i * \frac{m}{2} & \forall j \in \{1...m\} \\
& \sum_{1}^{j=m} dc_{i,j} = c_i & \forall i \in \{1...m\} \\
& consumed * \sum_{1}^{j=m} dc_{i,j} * census_j \leq stor_i & \forall i \in \{1...m\} \\
& dc_{i,i} = c_i & \forall i \in \{1...m\}
\end{aligned}
$$

# 4. Code

```
In [3]: using JuMP, Cbc
```

Data import

```
In [4]: raw = readcsv("data.csv")
        (c, n) = size(raw)

        m = c-1
        n_metro = 1:m
        readFile = 2:c

        metro = raw[readFile, 2][:]
        census = raw[readFile, 3][:]
        lat = raw[readFile, 4][:]
        long = raw[readFile, 5][:]
        fixed = raw[readFile, 6][:]
        cooling = raw[readFile, 7][:]

        for i = n_metro
            lat[i] =convert(Float64,  lat[i])
            long[i] =convert(Float64,  long[i])
        end
```

Helper Functions

```
In [5]: # haversine function for distance calculation
        haversine(lat1, lon1, lat2, lon2) = 2 * 6372.8 * asin(sqrt(sind((lat2 - lat1) / 2) ^ 2
                sind((lon2 - lon1) / 2) ^ 2))

        # data center serves cities within 800 miles of each other
        distance_limit = 800
        # initializing empty matrices
        dist = zeros(m,m)
        set = zeros(m,m)

        # set of distances from city i to j
        for i = n_metro
            for j = n_metro
                dist[i, j] = haversine(lat[i], long[i], lat[j], long[j])
            end
        end

        # set of cities within service limit
        for i = n_metro
            for j = n_metro
                if dist[i,j] < distance_limit
                    set[i,j] = 1
                end
            end
        end
```

```
In [6]: # print output with labels corresponding to the map.
        function printDataCenter(vals::Array{Float64,1})
            # label markers that match the maps produced by Batchgeo
            Letter = ["A","B","C","D","E","F","G","H","I","J","K","L","M"]
            count = 1;
            a = vals
            println("Data center locations: ")
            for i in 1:length(a)
                if (vals[i] == 1)
                    println(Letter[count], ": ", metro[i], " area.")
                    count = count+1
                end
            end
        end;
```

```
In [7]: # applicable to model 1
        function printStorageAtDC(storage::Array{Float64,1})
             for i in 1:length(storage)
                if (storage[i] > 0)
                    println("The ", metro[i], " data center is allocated ", storage[i], " TB o
                end
            end
        end;
```

```
In [8]: # applicable to models 2 and 3 to print cities j served by data center i.
        function printCityServed(vals::Array{Float64,2})
            for i = 1:m
            # a diagonal value of 1 indicates a data center is located in city i
            if (vals[i,i] == 1)
                println( "The data center in the [", metro[i], "] area serves: ")
                for j = 1:m
                    if (vals[i,j]==1)
                        println("    - ", metro[j])
                    end
                end
                println()
            end
        end
    end;
```

Model 0.0

```
In [9]: mod0 = Model(solver=CbcSolver())

        # selected cities
        @variable(mod0, c[1:m], Bin)

        # ensures each city is within 800 kilometers of a data center.
        @constraint(mod0, set * c .>= ones(m))

        # minimize total number of data centers
        @objective(mod0, Min, sum(c))

        solve(mod0)

        Mod0 = getvalue(c);
```

Storing result of Model 0.0 for use in Model 1.0

```
In [10]: # stores the value of c
         selected = zeros(m,m)
         for i = 1:m
             selected = getvalue(c)
         end

         # determine which cities each selected data center serves
         sel_set = zeros(m,m)
         for i = 1:m
             if selected[i] == 1
                 for j = 1:m
                     sel_set[i,j] = set[i,j]
                 end
             end
         end
```

Model 1.0

```
In [11]: mod1 = Model(solver=CbcSolver())

         # max storage size in TB
         max = 10^6

         # assumed per person data consumption. Unit TB.
         consumed = 0.001

         # storage size at each data center
         @variable(mod1, stor[1:m] >= 0)

         # initalizing an empty expression
         @expression(mod1, cost, 0)

         # binary upper bound
         @constraint(mod1, stor .<= selected * max)

         # expression for cooling cost with respect to storage size
         for i = 1:m
             @expression(mod1, cost, cost + (cooling[i] * stor[i]))
         end

         # minimal storage at each data center that would meet the demand of all cities.
         @constraint(mod1, [i=1:m], sum(consumed * sel_set[i,j] * census[j]  for j=1:m) <= sto

         @objective(mod1, Min, cost)

         solve(mod1)

         cost2 = getobjectivevalue(mod1);
         storage = getvalue(stor);
```

Model 2.0

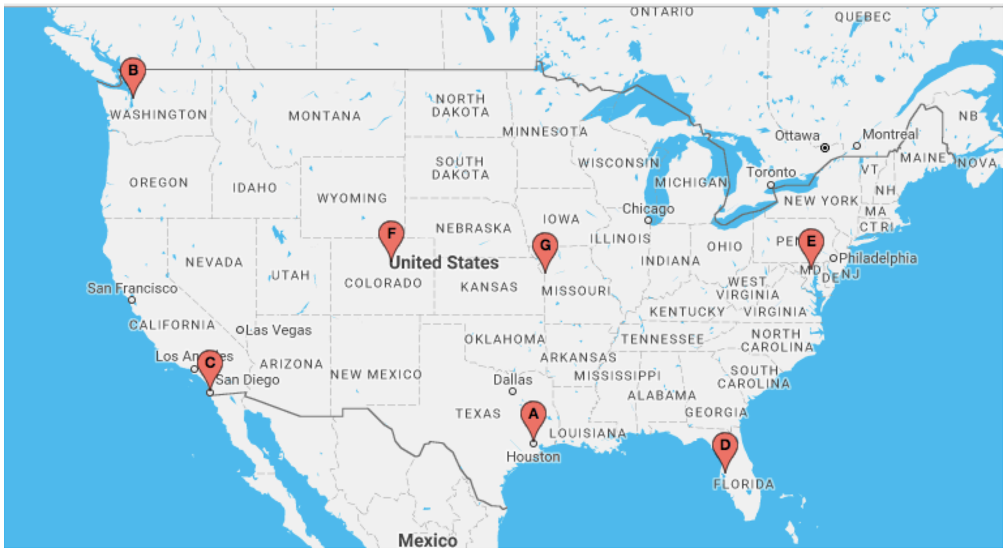
```
In [12]: # ****************************DO Not Run This Block of Code More Than Once**********
         # setting a large fixed cost multiplier for Model 2.0 and 3.0.


         multiplier = 1e5;
         fixed = fixed * multiplier;

In [13]: mod2 = Model(solver=CbcSolver())

         # max storage
         max = 10^6

         # assumed per person data consumption. Unit TB.
         consumed = 0.001
```

```
        vec = ones(m)

        lambda = 1

        @variable(mod2, c[1:m], Bin)

        # set of data centers i that serves cities j
        @variable(mod2, dc[1:m,1:m], Bin)

        # ensures each city is within 800 kilometers of a data center
        @constraint(mod2, set * c .>= vec)

        # fixed cost
        @expression(mod2, fcost, (0 * cooling + fixed)' * c)

        # each data center location can serve no more than m/2 cities
        @constraint(mod2, [i in 1:m], sum(dc[i,:]) <= c[i]*(m/2))

        # each city should be served
        @constraint(mod2, [j in 1:m], sum(dc[:,j]) == 1)

        # ensure 1) each data center i in city i serves itself and 2) minimum distance of 0 i
        # for cities that do not have datacenters in its location
        for i=1:m
            @constraint(mod2, dc[i,i] == c[i])
        end

        # calculates sum of distances between data center i and city j
        @expression(mod2, distance, sum(dist[i,j] * dc[i,j] for i=1:m, j=1:m))

        @objective(mod2, Min, lambda * fcost + distance)

        solve(mod2)

        Mod2 = getvalue(c);
        fixed_cost2 = getvalue(fcost);
        dist2 = getvalue(distance);
        dc2 = getvalue(dc);
```

Model 3.0

```
In [14]: mod3 = Model(solver=CbcSolver())

         vec = ones(m)

         # max storage
         max = 10^6
```

```
# assumed per person data consumption. Unit TB.
consumed = 0.001

@variable(mod3, c[1:m], Bin)

# storage size at each data center
@variable(mod3, stor[1:m] >= 0)

# set of data centers i that serves cities j
@variable(mod3, dc[1:m, 1:m], Bin)

# ensures each city is within 800 kilometers of a data center.
@constraint(mod3, set * c .>= vec)

# each data center location can serve no more than m/2 cities
@constraint(mod3, [i in 1:m], sum(dc[i,:]) <= c[i]*(m/2))

# each city should be served
@constraint(mod3, [j in 1:m], sum(dc[:,j]) == 1)

# ensure 1) each data center i in city i serves itself and 2) minimum distance of 0 i
# for cities that do not have datacenters in its location
for i=1:m
    @constraint(mod3, dc[i,i] == c[i])
end

# binary upper bound
@constraint(mod3, stor .<= c * max)

# initalizing an empty expression
@expression(mod3, vcost, 0)

# expression for cooling cost with respect to storage size
for i = 1:m
    @expression(mod3, vcost, vcost + (cooling[i] * stor[i]))
end

# fixed cost
@expression(mod3, fcost, (0 * cooling + fixed)' * c)

# minimal storage at each data center that would meet the demand of all cities.
@constraint(mod3, [i=1:m], sum(consumed * dc[i,j] * census[j]  for j=1:m) <= stor[i])

# ensures data center serves the city it's built in, or doesn't serve itself if there
@expression(mod3, distance, sum(dist[i,j] * dc[i,j] for i=1:m, j=1:m))

@objective(mod3, Min, vcost + fcost + distance)
```



alt text

```
    solve(mod3)

    Mod3 = getvalue(c);
    var_cost3 = getvalue(vcost);
    fix_cost3 = getvalue(fcost);
    dist3 = getvalue(distance);
    dc3 = getvalue(dc);
```

# 5. Results and Discussion We exported our result to Batchgeo and created the below maps to indicate data center locations.

Model 0.0

The above map indicates the set of cities where a data center should be built when considering the distance limit alone. In this simple model, we see an even scatter of data center locations across continental US, which is sensible from a distance perspective. Most of US population are concentrated on the coasts and in the south, so data centers constrained by distance are scattered in those regions. This map roughly resembles image 1 from our introduction.

We did expect to see data centers in northern states, but because this model did not involve cooling cost, the result did not meet this expectation.

This model nevertheless is not grounded in reality because data centers are rarely built based on distance. Therefore, we move on to Models 2 and 3 to incorporate further constraints.

Model 1.0

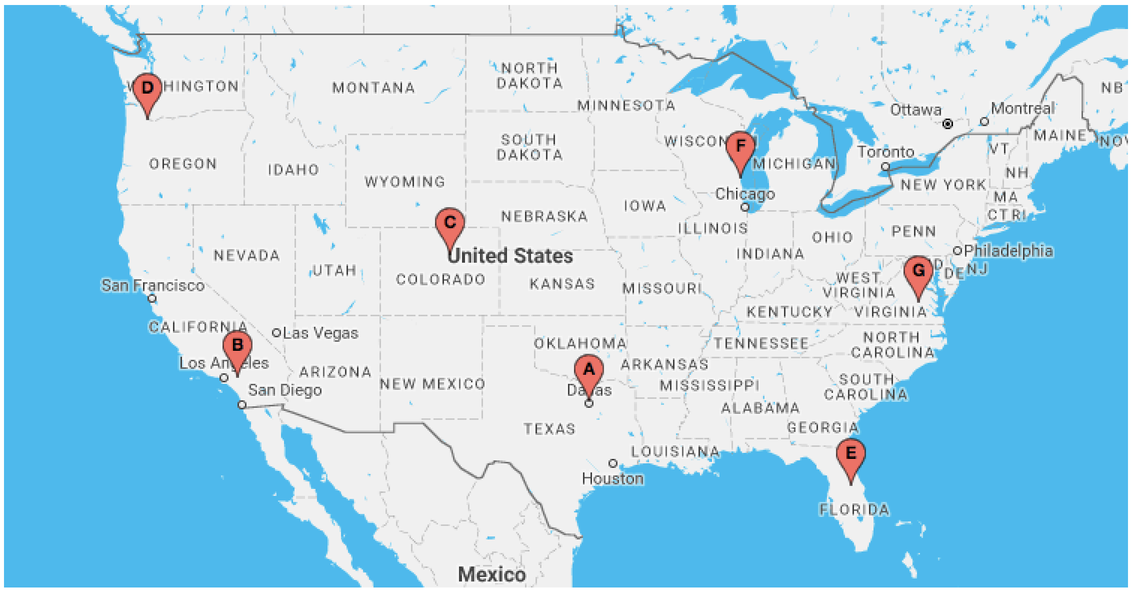
```
In [16]: printStorageAtDC(storage)

The Houston-The Woodlands-Sugar Land, TX data center is allocated 21187.071999999996 TB of sto
The Seattle-Tacoma-Bellevue, WA data center is allocated 5924.659 TB of storage.
The San Diego-Carlsbad, CA data center is allocated 35801.878000000004 TB of storage.
The Tampa-St. Petersburg-Clearwater, FL data center is allocated 20265.448999999997 TB of stora
The Baltimore-Columbia-Towson, MD data center is allocated 62670.787000000004 TB of storage.
The Denver-Aurora-Lakewood, CO data center is allocated 3837.959 TB of storage.
The Kansas City, MO-KS data center is allocated 33877.093 TB of storage.
```

In addition to Model 0, we wanted to see how much storage space each data center would receive in terms of demand. Each data center is bounded by a maximum storage limit and must at least meet customer demand.

The result of this trivial model shows that storage allocation is directly scaled to population. Therefore, once again, this model lacks realness because there are factors other than customer population for demand. We include this model mainly as a logic and functionality check prior to increasing complexity.

Model 2.0

```
In [17]: display("image/png", read("Model2.png"))
         printDataCenter(Mod2)
         println()
         println("Overall fixed cost for all data centers is \$", fixed_cost2, ".")
         println("Shortest distance between data centers and all its serviced cities is ", dis
         println("-----------------------------------------------------------------------------
         printCityServed(dc2)
```



```
Data center locations:
A: Dallas-Fort Worth-Arlington, TX area.
B: Riverside-San Bernardino-Ontario, CA area.
C: Denver-Aurora-Lakewood, CO area.
D: Portland-Vancouver-Hillsboro, OR-WA area.
E: Orlando-Kissimmee-Sanford, FL area.
F: Milwaukee-Waukesha-West Allis, WI area.
```

```
G: Richmond, VA area.

Overall fixed cost for all data centers is $1.50983e10.
Shortest distance between data centers and all its serviced cities is 19073.22181056693 kilome
--------------------------------------------------------------------------------------------------------
The data center in the [Dallas-Fort Worth-Arlington, TX] area serves:
   - Dallas-Fort Worth-Arlington, TX
   - Houston-The Woodlands-Sugar Land, TX
   - San Antonio-New Braunfels, TX
   - Austin-Round Rock, TX
   - Memphis, TN-MS-AR
   - Oklahoma City, OK
   - New Orleans-Metairie, LA

The data center in the [Riverside-San Bernardino-Ontario, CA] area serves:
   - Los Angeles-Long Beach-Anaheim, CA
   - San Francisco_Oakland_Hayward, CA
   - Phoenix-Mesa-Scottsdale, AZ
   - Riverside-San Bernardino-Ontario, CA
   - San Diego-Carlsbad, CA
   - Sacramentoosevillerden-Arcade, CA
   - Las Vegas-Henderson-Paradise, NV
   - San Jose-Sunnyvale-Santa Clara, CA

The data center in the [Denver-Aurora-Lakewood, CO] area serves:
   - Denver-Aurora-Lakewood, CO
   - Salt Lake City, UT

The data center in the [Portland-Vancouver-Hillsboro, OR-WA] area serves:
   - Seattle-Tacoma-Bellevue, WA
   - Portland-Vancouver-Hillsboro, OR-WA

The data center in the [Orlando-Kissimmee-Sanford, FL] area serves:
   - Miami-Fort Lauderdale-West Palm Beach, FL
   - Atlanta-Sandy Springs-Roswell, GA
   - Tampa-St. Petersburg-Clearwater, FL
   - Orlando-Kissimmee-Sanford, FL
   - Jacksonville, FL
   - Birmingham-Hoover, AL

The data center in the [Milwaukee-Waukesha-West Allis, WI] area serves:
   - Chicago-Naperville-Elgin, IL-IN-WI
   - Detroit-Warren-Dearborn, MI
   - Minneapolis-St. Paul-Bloomington, MN-WI
   - St. Louis, MO-IL
   - Cincinnati, OH-KY-IN
   - Cleveland-Elyria, OH
   - Kansas City, MO-KS
   - Columbus, OH
   - Indianapolis-Carmel-Anderson, IN
   - Nashville-Davidsonurfreesbororanklin, TN
   - Milwaukee-Waukesha-West Allis, WI
   - Louisville/Jefferson County, KY-IN

The data center in the [Richmond, VA] area serves:
   - New York-Newark-Jersey City, NY-NJ-PA
   - Philadelphia-Camden-Wilmington, PA-NJ-DE-MD
   - Washington-Arlington-Alexandria, DC-VA-MD-WV
   - Boston-Cambridge-Newton, MA-NH
   - Baltimore-Columbia-Towson, MD
   - Pittsburgh, PA
   - Charlotte-Concord-Gastonia, NC-SC
   - Virginia Beach-Norfolk-Newport News, VA-NC
   - Providence-Warwick, RI-MA
   - Richmond, VA
   - Hartford-West Hartford-East Hartford, CT
   - Raleigh, NC
   - Buffalo-Cheektowaga-Niagara Falls, NY
```

The above map shows the set of cities where a data center should be built when taken into the account of fixed cost and service distance. We wanted to make sure that each city is within 800 kilometers to a data center, however, taking into account the fixed costs and the distance from data center towards the server cities. The cities might not be served by the closest data center due to variations in cost at each data center.

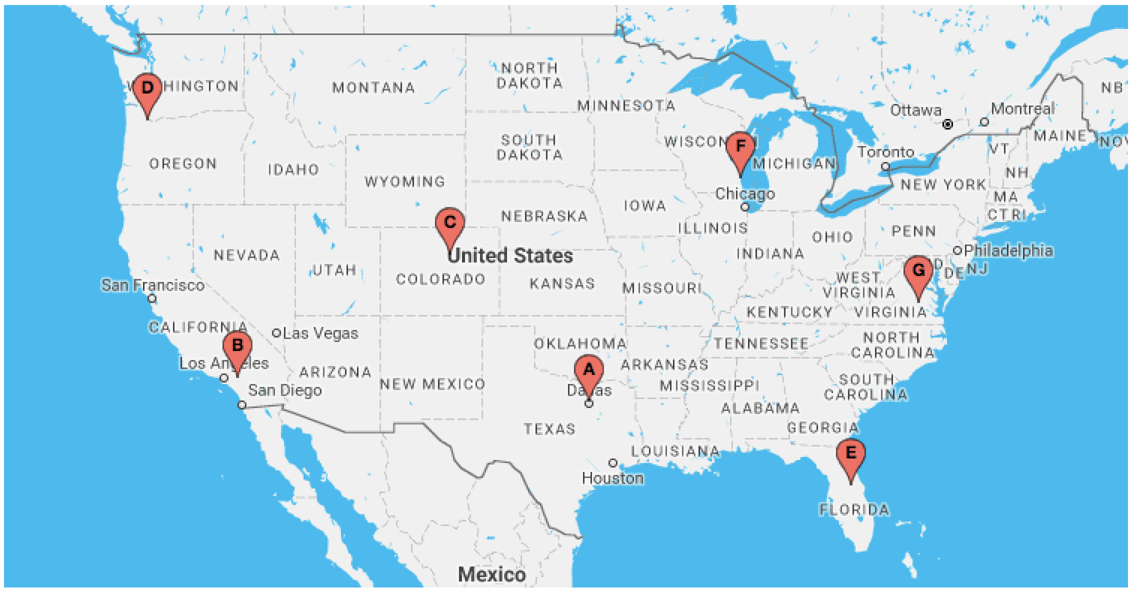
Although we equally weighted cost and distance, the cost had a larger affect on our objective because cost is 10^5 times greater than distance

Comparing data center allocation between Models 0 and 2: * The Houston center moved to Dallas. * The Kansas City center moved to Milwaukee. * The Tampa center moved to Orlando. * The Seattle center moved to Portland.

This change in allocation is interesting because the overall data center spread moved towards central US. This corresponds to the objective of minimizing overall service distance. In terms of cities served, it appears that the data centers mostly serves regional cities. For example, the Richmond center serves all mid Atlantic cities and the Milwaukee center serves all midwest cities. We expect to see a different result with the addition of cooling cost in Model 3.

Model 3.0

```
In [18]: display("image/png", read("Model3.png"))
         printDataCenter(Mod3)
         println()
         println("Overall fixed cost for all data centers is \$", fix_cost3, ".")
         println("Overall variable cost for all data centers is \$", var_cost3, ".")
         println("Shortest distance between data centers and all its serviced cities is ", dis
         println("-----------------------------------------------------------------------------
         printCityServed(dc3)
```



```
Data center locations:
A: Dallas-Fort Worth-Arlington, TX area.
B: Riverside-San Bernardino-Ontario, CA area.
C: Denver-Aurora-Lakewood, CO area.
D: Portland-Vancouver-Hillsboro, OR-WA area.
E: Orlando-Kissimmee-Sanford, FL area.
F: Milwaukee-Waukesha-West Allis, WI area.
G: Richmond, VA area.

Overall fixed cost for all data centers is $1.50983e10.
Overall variable cost for all data centers is $1.01885791483876e9.
Shortest distance between data centers and all its serviced cities is 77174.17372286758 kilome
-------------------------------------------------------------------------------------------------------
The data center in the [Dallas-Fort Worth-Arlington, TX] area serves:
   - Dallas-Fort Worth-Arlington, TX

The data center in the [Riverside-San Bernardino-Ontario, CA] area serves:
   - Riverside-San Bernardino-Ontario, CA

The data center in the [Denver-Aurora-Lakewood, CO] area serves:
   - Denver-Aurora-Lakewood, CO
   - Kansas City, MO-KS
   - Las Vegas-Henderson-Paradise, NV
   - Columbus, OH
   - Indianapolis-Carmel-Anderson, IN
   - San Jose-Sunnyvale-Santa Clara, CA
   - Austin-Round Rock, TX
   - Nashville-Davidsonurfreesbororanklin, TN
   - Virginia Beach-Norfolk-Newport News, VA-NC
   - Providence-Warwick, RI-MA
   - Jacksonville, FL
   - Memphis, TN-MS-AR
   - Oklahoma City, OK
   - Louisville/Jefferson County, KY-IN
   - New Orleans-Metairie, LA
   - Hartford-West Hartford-East Hartford, CT
   - Raleigh, NC
   - Salt Lake City, UT
   - Birmingham-Hoover, AL
   - Buffalo-Cheektowaga-Niagara Falls, NY

The data center in the [Portland-Vancouver-Hillsboro, OR-WA] area serves:
   - Portland-Vancouver-Hillsboro, OR-WA

The data center in the [Orlando-Kissimmee-Sanford, FL] area serves:
   - New York-Newark-Jersey City, NY-NJ-PA
   - Los Angeles-Long Beach-Anaheim, CA
   - Chicago-Naperville-Elgin, IL-IN-WI
   - Houston-The Woodlands-Sugar Land, TX
   - Philadelphia-Camden-Wilmington, PA-NJ-DE-MD
   - Washington-Arlington-Alexandria, DC-VA-MD-WV
   - Miami-Fort Lauderdale-West Palm Beach, FL
   - Atlanta-Sandy Springs-Roswell, GA
   - Boston-Cambridge-Newton, MA-NH
   - San Francisco_Oakland_Hayward, CA
   - Phoenix-Mesa-Scottsdale, AZ
   - Detroit-Warren-Dearborn, MI
   - Seattle-Tacoma-Bellevue, WA
   - Minneapolis-St. Paul-Bloomington, MN-WI
   - San Diego-Carlsbad, CA
   - Tampa-St. Petersburg-Clearwater, FL
   - St. Louis, MO-IL
   - Baltimore-Columbia-Towson, MD
   - Pittsburgh, PA
   - Charlotte-Concord-Gastonia, NC-SC
   - San Antonio-New Braunfels, TX
   - Orlando-Kissimmee-Sanford, FL
   - Sacramentoosevillerden-Arcade, CA
   - Cincinnati, OH-KY-IN
   - Cleveland-Elyria, OH

The data center in the [Milwaukee-Waukesha-West Allis, WI] area serves:
   - Milwaukee-Waukesha-West Allis, WI

The data center in the [Richmond, VA] area serves:
   - Richmond, VA
```

The above map shows the set of cities hosting data centers when taken into the account of fixed cost, variable cost, and service distance. When compared with Model 2, the data centers are still located in the same cities. However, there is a substantial difference in cities served by each data center. Notably, the data centers no longer serve cities in its region. For example, the FL center also services western states such as AZ and CA as well as Midwestern states such as MI and OH. The variable costs have a bigger effect on the objective function than distance. Thus, we can see some cities served by a data center that is located far from the city, because the variable costs of the further data center can be significantly lower than the variable costs of a closer data center

Upon closer inspection, it's also clear that in Model 3, cities served per data center becomes polarized. This is characterized by the CO and FL centers servicing the majority of cities while the other data centers only serve its own city's population. We did not expect this result, but it is sensible.

# 6. Conclusion

This report investigates the placement of data centers approximated by the distance from servers to consumers. We were able to demonstrate the best placement for new data centers given a required maximum distance (approximating latency), as well as the approximate data storage size. The program could be used in the following ways:

1. Discover the best places to put caching servers in the US to reduce latency (using a small latency or distance).
2. Discover the best places to put CDN servers in the US to reduce latency (where we'd set a larger latency, or distance).
3. Locate the best locations to open up new data centers, especially those moving off of AWS and looking to set up their own server farms.
4. Research and compare the current state of major data center hubs and compare it with what would be optimal.

Moving forward in the project would be to continue to add more functionality into the placement of data centers. Adding increased complexity in the calculations of costs (such as detailed taxes and cost of talent in that area, such as Network Engineers in the bay area would be more expensive than outside of Boston), as well as the ability to find optimal cities given data centers already in place. For increased accuracy, other factors should be included rather than just population to estimate demand. Specifically, we would distinguish individual users from corporate and academic users as well as incorporating existing internet traffic. Lastly expanding these calculations to beyond the US would allow more areas to benefit from these algorithms.

# Appendix

Below is the data used to calculate our data, based on (in order) the city, census, longitude, latitude, populations, and fixed costs of the cities, where the "fixed costs" were quasi random based off real estimations.

```
In [19]: println("City  #: population latitude longitude  cooling   fixed")
         for i = 1:m
             @printf("City %2d: ",i)
             @printf("%9.4d | ", census[i]);
             @printf("%6.2d | ", lat[i])
             @printf("%7.2d | ", long[i])
```

```
            @printf("%6.2d | ",  cooling[i])
            @printf("%6.2d |\n", fixed[i])
        end
```

```
City  #: population latitude longitude  cooling   fixed
City  1:  19949502 |     41 |      -74 |   5929 | 2418400000 |
City  2:  13131431 |     34 |     -118 |   6595 | 2327000000 |
City  3:   9537289 |     42 |      -88 |   5812 | 2206700000 |
City  4:   6810913 |     33 |      -97 |   6722 | 2027200000 |
City  5:   6313158 |     30 |      -95 |   7024 | 2244500000 |
City  6:   6034678 |     40 |      -75 |   6005 | 2751500000 |
City  7:   5949859 |     39 |      -77 |   6109 | 2048200000 |
City  8:   5828191 |     26 |      -80 |   7421 | 2448200000 |
City  9:   5522942 |     34 |      -84 |   6625 | 2386400000 |
City 10:   4684299 |     42 |      -71 |   5764 | 2122900000 |
City 11:   4516276 |     38 |     -122 |   5929 | 2418400000 |
City 12:   4398762 |     33 |     -112 |   5812 | 2206700000 |
City 13:   4380878 |     34 |     -117 |   6722 | 2027200000 |
City 14:   4294983 |     42 |      -83 |   7024 | 2244500000 |
City 15:   3610105 |     48 |     -122 |   6005 | 2751500000 |
City 16:   3459146 |     45 |      -93 |   6005 | 2751500000 |
City 17:   3211252 |     33 |     -117 |   6109 | 2048200000 |
City 18:   2870569 |     28 |      -82 |   7421 | 2448200000 |
City 19:   2810056 |     39 |      -90 |   6625 | 2386400000 |
City 20:   2770738 |     39 |      -77 |   5764 | 2122900000 |
City 21:   2697476 |     40 |     -105 |   5929 | 2418400000 |
City 22:   2360867 |     40 |      -80 |   6595 | 2327000000 |
City 23:   2335358 |     35 |      -81 |   6109 | 2048200000 |
City 24:   2314554 |     46 |     -123 |   7421 | 2448200000 |
City 25:   2277550 |     29 |      -98 |   6625 | 2386400000 |
City 26:   2267846 |     29 |      -81 |   5764 | 2122900000 |
City 27:   2215770 |     39 |     -121 |   5929 | 2418400000 |
City 28:   2137406 |     39 |      -85 |   6595 | 2327000000 |
City 29:   2064725 |     42 |      -82 |   5812 | 2206700000 |
City 30:   2054473 |     39 |      -95 |   6109 | 2048200000 |
City 31:   2027868 |     36 |     -115 |   7421 | 2448200000 |
City 32:   1967066 |     40 |      -83 |   6625 | 2386400000 |
City 33:   1953961 |     40 |      -86 |   5764 | 2122900000 |
City 34:   1919641 |     37 |     -122 |   6722 | 2027200000 |
City 35:   1883051 |     30 |      -98 |   7024 | 2244500000 |
City 36:   1757912 |     36 |      -87 |   6005 | 2751500000 |
City 37:   1707369 |     37 |      -76 |   6595 | 2327000000 |
City 38:   1604291 |     42 |      -71 |   5812 | 2206700000 |
City 39:   1569659 |     43 |      -88 |   6722 | 2027200000 |
City 40:   1394624 |     30 |      -82 |   7024 | 2244500000 |
City 41:   1341746 |     35 |      -90 |   5929 | 2418400000 |
City 42:   1319677 |     35 |      -98 |   6595 | 2327000000 |
City 43:   1262261 |     38 |      -86 |   5812 | 2206700000 |
City 44:   1245764 |      38 |     -77 |   6722 | 2027200000 |
City 45:   1240977 |      30 |     -90 |   7024 | 2244500000 |
City 46:   1215211 |      42 |     -73 |   6005 | 2751500000 |
City 47:   1214516 |      36 |     -79 |   6109 | 2048200000 |
City 48:   1140483 |      41 |    -112 |   7421 | 2448200000 |
City 49:   1140300 |      34 |     -87 |   6625 | 2386400000 |
City 50:   1134155 |      43 |     -79 |   5764 | 2122900000 |
```