In [1]:
```
from gerrychain import Graph
```

In [2]:
```
# Read Alabama county graph from the json file "AL_county.json"
filepath = 'C:\\Users\\blrod\\Downloads\\districting-data-2020-county\\'
filename = 'ALL_county.json'

# GerryChain has a built-in function for reading graphs of this type:
G = Graph.from_json( filepath + filename )
```

In [3]:
```
# For each node, print the node #, county name, population, and lat-long coordinates
for node in G.nodes:
    name = G.nodes[node]["NAME20"]
    population = G.nodes[node]['P0010001']
    G.nodes[node]['TOTPOP'] = population

    # query lat and long coordinates
    G.nodes[node]['C_X'] = G.nodes[node]['INTPTLON20']  #longitude of county's center
    G.nodes[node]['C_Y'] = G.nodes[node]['INTPTLAT20']  #latitude of county's center

    print("Node",node,"is",name,"County, which has population",population,"and is centered at (",G.nodes[node]['C_X'],",",G.nodes
```

```
Node 0 is Shelby County, which has population 223024 and is centered at ( -086.6780894 , +33.2630428 )
Node 1 is Dallas County, which has population 38462 and is centered at ( -087.1143600 , +32.3335263 )
Node 2 is Pickens County, which has population 19123 and is centered at ( -088.0968644 , +33.2968003 )
Node 3 is Lauderdale County, which has population 93564 and is centered at ( -087.6509966 , +34.9041221 )
Node 4 is Cleburne County, which has population 15056 and is centered at ( -085.5161261 , +33.6719637 )
Node 5 is Barbour County, which has population 25223 and is centered at ( -085.4051035 , +31.8702531 )
Node 6 is Geneva County, which has population 26659 and is centered at ( -085.8210224 , +31.0923822 )
Node 7 is Dale County, which has population 49326 and is centered at ( -085.6094760 , +31.4306536 )
Node 8 is Tallapoosa County, which has population 41311 and is centered at ( -085.7996176 , +32.8633076 )
Node 9 is Clarke County, which has population 23087 and is centered at ( -087.8186244 , +31.6855211 )
Node 10 is Houston County, which has population 107202 and is centered at ( -085.2964111 , +31.1581831 )
Node 11 is Washington County, which has population 15388 and is centered at ( -088.2124041 , +31.4085035 )
Node 12 is Madison County, which has population 388153 and is centered at ( -086.5510802 , +34.7642377 )
Node 13 is Crenshaw County, which has population 13194 and is centered at ( -086.3200384 , +31.7303106 )
Node 14 is Calhoun County, which has population 116441 and is centered at ( -085.8279089 , +33.7705162 )
Node 15 is Lawrence County, which has population 33073 and is centered at ( -087.3218651 , +34.5297760 )
Node 16 is Morgan County, which has population 123421 and is centered at ( -086.8464021 , +34.4544844 )
Node 17 is Lamar County, which has population 13972 and is centered at ( -088.0874309 , +33.7870852 )
Node 18 is Russell County, which has population 59183 and is centered at ( -085.1869798 , +32.2898113 )
Node 19 is Franklin County, which has population 32113 and is centered at ( -087.8428144 , +34.4419892 )
Node 20 is Conecuh County, which has population 11597 and is centered at ( -086.9887221 , +31.4309257 )
Node 21 is Elmore County, which has population 87977 and is centered at ( -086.1427347 , +32.5972290 )
Node 22 is Jefferson County, which has population 674721 and is centered at ( -086.8965359 , +33.5534439 )
Node 23 is Walker County, which has population 65342 and is centered at ( -087.3010936 , +33.7915581 )
Node 24 is Randolph County, which has population 21967 and is centered at ( -085.4640637 , +33.2964614 )
Node 25 is Montgomery County, which has population 228954 and is centered at ( -086.2044615 , +32.2028812 )
Node 26 is Bibb County, which has population 22293 and is centered at ( -087.1271475 , +33.0158929 )
Node 27 is Etowah County, which has population 103436 and is centered at ( -086.0342629 , +34.0476407 )
Node 28 is Chilton County, which has population 45014 and is centered at ( -086.7266071 , +32.8540514 )
Node 29 is Coffee County, which has population 53465 and is centered at ( -085.9896022 , +31.4022580 )
Node 30 is Covington County, which has population 37570 and is centered at ( -086.4487206 , +31.2439873 )
Node 31 is Henry County, which has population 17146 and is centered at ( -085.2399712 , +31.5169779 )
Node 32 is Clay County, which has population 14236 and is centered at ( -085.8635254 , +33.2703999 )
Node 33 is Marengo County, which has population 19323 and is centered at ( -087.7910910 , +32.2475911 )
Node 34 is DeKalb County, which has population 71608 and is centered at ( -085.8040207 , +34.4609148 )
Node 35 is Cherokee County, which has population 24971 and is centered at ( -085.6542417 , +34.0695153 )
Node 36 is Hale County, which has population 14785 and is centered at ( -087.6230608 , +32.7527958 )
Node 37 is Perry County, which has population 8511 and is centered at ( -087.2938269 , +32.6390053 )
Node 38 is Colbert County, which has population 57227 and is centered at ( -087.8014569 , +34.7031120 )
Node 39 is Greene County, which has population 7730 and is centered at ( -087.9642005 , +32.8444965 )
Node 40 is Butler County, which has population 19051 and is centered at ( -086.6819689 , +31.7516670 )
Node 41 is Lee County, which has population 174241 and is centered at ( -085.3530477 , +32.6040644 )
```

```
Node 42 is Mobile County, which has population 414809 and is centered at ( -088.1965682 , +30.6845725 )
Node 43 is Fayette County, which has population 16321 and is centered at ( -087.7642923 , +33.7161568 )
Node 44 is Chambers County, which has population 34772 and is centered at ( -085.3940321 , +32.9155039 )
Node 45 is Tuscaloosa County, which has population 227036 and is centered at ( -087.5227834 , +33.2902197 )
Node 46 is Wilcox County, which has population 10600 and is centered at ( -087.3049349 , +31.9900824 )
Node 47 is Marshall County, which has population 97612 and is centered at ( -086.3216681 , +34.3095637 )
Node 48 is Escambia County, which has population 36757 and is centered at ( -087.1684097 , +31.1222867 )
Node 49 is Limestone County, which has population 103570 and is centered at ( -086.9813995 , +34.8102387 )
Node 50 is Blount County, which has population 59134 and is centered at ( -086.5664400 , +33.9773575 )
Node 51 is Monroe County, which has population 19772 and is centered at ( -087.3832656 , +31.5803324 )
Node 52 is Marion County, which has population 29341 and is centered at ( -087.8815510 , +34.1382194 )
Node 53 is Lowndes County, which has population 10311 and is centered at ( -086.6505859 , +32.1478880 )
Node 54 is Coosa County, which has population 10387 and is centered at ( -086.2434818 , +32.9314453 )
Node 55 is Pike County, which has population 33009 and is centered at ( -085.9416076 , +31.7986533 )
Node 56 is Sumter County, which has population 12345 and is centered at ( -088.2000571 , +32.5974811 )
Node 57 is Winston County, which has population 23540 and is centered at ( -087.3653458 , +34.1545665 )
Node 58 is Talladega County, which has population 82149 and is centered at ( -086.1759302 , +33.3693135 )
Node 59 is Jackson County, which has population 52579 and is centered at ( -085.9800556 , +34.7641140 )
Node 60 is Baldwin County, which has population 231767 and is centered at ( -087.7460666 , +30.6592183 )
Node 61 is Bullock County, which has population 10357 and is centered at ( -085.7172613 , +32.1017589 )
Node 62 is Autauga County, which has population 58805 and is centered at ( -086.6464395 , +32.5322367 )
Node 63 is Macon County, which has population 19532 and is centered at ( -085.6928870 , +32.3870267 )
Node 64 is St. Clair County, which has population 91103 and is centered at ( -086.3113273 , +33.7194907 )
Node 65 is Choctaw County, which has population 12665 and is centered at ( -088.2488894 , +31.9909539 )
Node 66 is Cullman County, which has population 87866 and is centered at ( -086.8692666 , +34.1319229 )
```

In [4]:
```
pip install geopy
```

```
Requirement already satisfied: geopy in c:\users\blrod\anaconda3\lib\site-packages (2.2.0)Note: you may need to restart the kern
el to use updated packages.

Requirement already satisfied: geographiclib<2,>=1.49 in c:\users\blrod\anaconda3\lib\site-packages (from geopy) (1.52)
```

In [5]:
```python
from geopy.distance import geodesic

# create distance dictionary
dist = { (i,j) : 0 for i in G.nodes for j in G.nodes }
for i in G.nodes:
    for j in G.nodes:
        loc_i = (G.nodes[i]['C_Y'],G.nodes[i]['C_X'])
        loc_j = (G.nodes[j]['C_Y'],G.nodes[j]['C_X'])
        dist[i,j] = geodesic(loc_i,loc_j).miles
```

In [6]:
```
# Let's impose a 1% population deviation (+/- 0.5%)
deviation = 0.01

import math
k = 7          # number of districts
total_population = sum(G.nodes[node]['TOTPOP'] for node in G.nodes)

L = math.ceil((1-deviation/2)*total_population/k)
U = math.floor((1+deviation/2)*total_population/k)
print("Using L =",L,"and U =",U,"and k =",k)
```

```
Using L = 714166 and U = 721342 and k = 7
```

In [7]:
```
import gurobipy as gp
from gurobipy import GRB

# create model
m =gp.Model()

# create x[i,j] variable which equals one when county i
#    is assigned to (the district centered at) county j
x =m.addVars( G.nodes, G.nodes, vtype=GRB.BINARY)
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2022-06-09
```

In [8]:
```
# objective is to minimize the moment of inertia: sum (d^2 * p * x over all i and j)
m.setObjective( gp.quicksum( dist[i,j] * dist[i,j] *G.nodes[i]['TOTPOP'] * x[i,j] for i in G.nodes for j in G.nodes ), GRB.MINIMI
```

In [9]:
```
# add constraints saying that each county i is assigned to one district
m.addConstrs( gp.quicksum( x[i,j] for j in G.nodes ) == 1 for i in G.nodes)

# add constraint saying there should be k district centers
m.addConstr( gp.quicksum( x[j,j] for j in G.nodes ) == k )

# add constraints that say: if j roots a district, then its population is between L and U.
m.addConstrs( gp.quicksum( G.nodes[i]['TOTPOP'] * x[i,j] for i in G.nodes ) >= L * x[j,j] for j in G.nodes )
m.addConstrs( gp.quicksum( G.nodes[i]['TOTPOP'] * x[i,j] for i in G.nodes ) <= U * x[j,j] for j in G.nodes )

# add coupling constraints saying that if i is assigned to j, then j is a center.
m.addConstrs( x[i,j] <= x[j,j] for i in G.nodes for j in G.nodes )
```

```
m.update()
```

In [10]:
```
# add contiguity constraints
import networkx as nx
DG = nx.DiGraph(G)

#add flow variables
f = m.addVars( DG.edges, G.nodes ) # f[i,j,v] = flow across arc (i,j) that is sent from source/root v

#add constraints saying that if node i is assigned to node j
# then node i must consume one unit of node j's flow
m.addConstrs( gp.quicksum( f[u,i,j] - f[i,u,j] for u in G.neighbors(i) ) == x[i,j] for i in G.nodes for j in G.nodes if i != j )

# add constraints saying that node i can recieve flow of type j
#only if node i is assigned to node j
M = G.number_of_nodes() - 1
m.addConstrs( gp.quicksum( f[u,i,j] for u in G.neighbors(i) ) <= M * x[i,j] for i in G.nodes for j in G.nodes if i != j )

#add constraints saying that j cannot recieve flow of its own type
m.addConstrs( gp.quicksum( f[u,j,j] for u in G.neighbors(j) ) == 0 for j in G.nodes )
```

```
Out[10]: {0: <gurobi.Constr *Awaiting Model Update*>,
 1: <gurobi.Constr *Awaiting Model Update*>,
 2: <gurobi.Constr *Awaiting Model Update*>,
 3: <gurobi.Constr *Awaiting Model Update*>,
 4: <gurobi.Constr *Awaiting Model Update*>,
 5: <gurobi.Constr *Awaiting Model Update*>,
 6: <gurobi.Constr *Awaiting Model Update*>,
 7: <gurobi.Constr *Awaiting Model Update*>,
 8: <gurobi.Constr *Awaiting Model Update*>,
 9: <gurobi.Constr *Awaiting Model Update*>,
 10: <gurobi.Constr *Awaiting Model Update*>,
 11: <gurobi.Constr *Awaiting Model Update*>,
 12: <gurobi.Constr *Awaiting Model Update*>,
 13: <gurobi.Constr *Awaiting Model Update*>,
 14: <gurobi.Constr *Awaiting Model Update*>,
 15: <gurobi.Constr *Awaiting Model Update*>,
 16: <gurobi.Constr *Awaiting Model Update*>,
 17: <gurobi.Constr *Awaiting Model Update*>,
 18: <gurobi.Constr *Awaiting Model Update*>,
 19: <gurobi.Constr *Awaiting Model Update*>,
 20: <gurobi.Constr *Awaiting Model Update*>,
 21: <gurobi.Constr *Awaiting Model Update*>,
 22: <gurobi.Constr *Awaiting Model Update*>,
 23: <gurobi.Constr *Awaiting Model Update*>,
 24: <gurobi.Constr *Awaiting Model Update*>,
 25: <gurobi.Constr *Awaiting Model Update*>,
 26: <gurobi.Constr *Awaiting Model Update*>,
 27: <gurobi.Constr *Awaiting Model Update*>,
 28: <gurobi.Constr *Awaiting Model Update*>,
 29: <gurobi.Constr *Awaiting Model Update*>,
 30: <gurobi.Constr *Awaiting Model Update*>,
 31: <gurobi.Constr *Awaiting Model Update*>,
 32: <gurobi.Constr *Awaiting Model Update*>,
 33: <gurobi.Constr *Awaiting Model Update*>,
 34: <gurobi.Constr *Awaiting Model Update*>,
 35: <gurobi.Constr *Awaiting Model Update*>,
 36: <gurobi.Constr *Awaiting Model Update*>,
 37: <gurobi.Constr *Awaiting Model Update*>,
 38: <gurobi.Constr *Awaiting Model Update*>,
 39: <gurobi.Constr *Awaiting Model Update*>,
 40: <gurobi.Constr *Awaiting Model Update*>,
```

```
41: <gurobi.Constr *Awaiting Model Update*>,
42: <gurobi.Constr *Awaiting Model Update*>,
43: <gurobi.Constr *Awaiting Model Update*>,
44: <gurobi.Constr *Awaiting Model Update*>,
45: <gurobi.Constr *Awaiting Model Update*>,
46: <gurobi.Constr *Awaiting Model Update*>,
47: <gurobi.Constr *Awaiting Model Update*>,
48: <gurobi.Constr *Awaiting Model Update*>,
49: <gurobi.Constr *Awaiting Model Update*>,
50: <gurobi.Constr *Awaiting Model Update*>,
51: <gurobi.Constr *Awaiting Model Update*>,
52: <gurobi.Constr *Awaiting Model Update*>,
53: <gurobi.Constr *Awaiting Model Update*>,
54: <gurobi.Constr *Awaiting Model Update*>,
55: <gurobi.Constr *Awaiting Model Update*>,
56: <gurobi.Constr *Awaiting Model Update*>,
57: <gurobi.Constr *Awaiting Model Update*>,
58: <gurobi.Constr *Awaiting Model Update*>,
59: <gurobi.Constr *Awaiting Model Update*>,
60: <gurobi.Constr *Awaiting Model Update*>,
61: <gurobi.Constr *Awaiting Model Update*>,
62: <gurobi.Constr *Awaiting Model Update*>,
63: <gurobi.Constr *Awaiting Model Update*>,
64: <gurobi.Constr *Awaiting Model Update*>,
65: <gurobi.Constr *Awaiting Model Update*>,
66: <gurobi.Constr *Awaiting Model Update*>}
```

In [11]:

```
# solve, making sure to set a 0.00% MIP gap tolerance(!)
m.Params.MIPGap = 0.0

m.optimize()
```

```
Set parameter MIPGap to value 0
Gurobi Optimizer version 9.5.0 build v9.5.0rc5 (win64)
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads
Optimize a model with 13602 rows, 27403 columns and 99280 nonzeros
Model fingerprint: 0xf2194a25
Variable types: 22914 continuous, 4489 integer (4489 binary)
Coefficient statistics:
  Matrix range     [1e+00, 7e+05]
  Objective range  [3e+06, 4e+10]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 7e+00]
Warning: Model contains large objective coefficients
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Presolve removed 575 rows and 1552 columns
Presolve time: 1.62s
Presolved: 13027 rows, 25851 columns, 95568 nonzeros
Variable types: 21445 continuous, 4406 integer (4406 binary)

Root relaxation: objective 5.572088e+09, 1832 iterations, 0.65 seconds (0.25 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 5.5721e+09    0  191          - 5.5721e+09      -     -    3s
     0     0 5.6869e+09    0  235          - 5.6869e+09      -     -    5s
     0     0 5.7372e+09    0  241          - 5.7372e+09      -     -    5s
     0     0 5.7373e+09    0  241          - 5.7373e+09      -     -    5s
     0     0 5.7836e+09    0  262          - 5.7836e+09      -     -    6s
     0     0 5.7971e+09    0  252          - 5.7971e+09      -     -    7s
     0     0 5.8001e+09    0  249          - 5.8001e+09      -     -    7s
     0     0 5.8013e+09    0  254          - 5.8013e+09      -     -    7s
     0     0 5.8014e+09    0  254          - 5.8014e+09      -     -    7s
     0     0 5.8014e+09    0  254          - 5.8014e+09      -     -    7s
     0     0 5.8829e+09    0  248          - 5.8829e+09      -     -    8s
     0     0 5.9159e+09    0  265          - 5.9159e+09      -     -    9s
     0     0 5.9177e+09    0  250          - 5.9177e+09      -     -    9s
     0     0 5.9177e+09    0  250          - 5.9177e+09      -     -    9s
     0     0 5.9513e+09    0  285          - 5.9513e+09      -     -   10s
     0     0 5.9702e+09    0  278          - 5.9702e+09      -     -   11s
     0     0 5.9707e+09    0  287          - 5.9707e+09      -     -   11s
     0     0 5.9708e+09    0  288          - 5.9708e+09      -     -   11s
```

```
     0     0 5.9709e+09    0  283          - 5.9709e+09      -     -   11s
     0     0 5.9709e+09    0  288          - 5.9709e+09      -     -   11s
     0     0 5.9788e+09    0  298          - 5.9788e+09      -     -   12s
     0     0 5.9792e+09    0  319          - 5.9792e+09      -     -   12s
     0     0 5.9792e+09    0  319          - 5.9792e+09      -     -   12s
     0     0 5.9900e+09    0  310          - 5.9900e+09      -     -   13s
     0     0 5.9912e+09    0  308          - 5.9912e+09      -     -   13s
     0     0 5.9912e+09    0  308          - 5.9912e+09      -     -   13s
     0     0 5.9990e+09    0  303          - 5.9990e+09      -     -   14s
     0     0 5.9998e+09    0  305          - 5.9998e+09      -     -   14s
     0     0 6.0001e+09    0  303          - 6.0001e+09      -     -   14s
     0     0 6.0002e+09    0  306          - 6.0002e+09      -     -   14s
     0     0 6.0002e+09    0  305          - 6.0002e+09      -     -   14s
     0     0 6.0017e+09    0  304          - 6.0017e+09      -     -   15s
     0     0 6.0019e+09    0  316          - 6.0019e+09      -     -   15s
     0     0 6.0020e+09    0  306          - 6.0020e+09      -     -   15s
     0     0 6.0028e+09    0  311          - 6.0028e+09      -     -   16s
     0     0 6.0028e+09    0  311          - 6.0028e+09      -     -   16s
     0     0 6.0032e+09    0  318          - 6.0032e+09      -     -   17s
     0     0 6.0032e+09    0  307          - 6.0032e+09      -     -   17s
     0     0 6.0036e+09    0  318          - 6.0036e+09      -     -   17s
     0     0 6.0036e+09    0  318          - 6.0036e+09      -     -   17s
     0     2 6.0036e+09    0  318          - 6.0036e+09      -     -   24s
     3     6 6.0329e+09    2  263          - 6.0182e+09      -   235   25s
    70    73 9.8600e+09   31   48          - 6.0182e+09      -   177   30s
   185   172 7.5032e+09   40   47          - 6.0250e+09      -   140   35s
   321   252 7.6037e+09   21   50          - 6.0331e+09      -   126   40s
   420   328 8.8213e+09   21   79          - 6.0632e+09      -   129   45s
   557   454 6.8133e+09   21   20          - 6.0882e+09      -   128   50s
   678   538 6.1615e+09    7  217          - 6.0897e+09      -   128   61s
   710   545 6.3491e+09   11   94          - 6.0897e+09      -   149   71s
   759   573 6.4377e+09   13  318          - 6.0897e+09      -   169   78s
   761   574 1.9902e+10   57  189          - 6.0897e+09      -   168   82s
   763   576 7.2703e+09   35  275          - 6.0897e+09      -   168   85s
   772   582 7.3059e+09   26  318          - 6.0897e+09      -   166   90s
   773   582 6.5776e+09   16  309          - 6.1022e+09      -   165   98s
   776   584 7.3281e+09   28  317          - 6.1255e+09      -   165  101s
   781   588 7.0491e+09    8  318          - 6.1374e+09      -   164  105s
   785   590 7.1643e+09   31  348          - 6.1463e+09      -   163  110s
   791   594 7.8185e+09   14  331          - 6.1547e+09      -   162  116s
   797   598 1.0393e+10   23  328          - 6.1689e+09      -   160  121s
   801   601 7.8684e+09   25  348          - 6.1711e+09      -   160  135s
```

803	602	7.3217e+09	28	348	-	6.1711e+09	-	159	149s
804	606	6.2006e+09	10	338	-	6.1712e+09	-	15.3	171s
806	607	6.2056e+09	11	300	-	6.1907e+09	-	15.9	184s
808	609	6.4718e+09	11	260	-	6.2084e+09	-	17.6	192s
812	611	6.3851e+09	12	267	-	6.2124e+09	-	20.1	196s
821	618	6.5045e+09	14	123	-	6.2228e+09	-	24.3	200s
845	638	6.6751e+09	20	144	-	6.2228e+09	-	30.7	205s
866	653	7.8057e+09	23	111	-	6.2228e+09	-	36.2	210s
907	674	7.1317e+09	28	163	-	6.2228e+09	-	40.5	215s
970	721	7.5499e+09	42	17	-	6.2228e+09	-	47.5	220s
1017	728	6.2518e+09	15	205	-	6.2500e+09	-	52.1	225s
1063	752	6.9282e+09	24	52	-	6.2500e+09	-	57.8	230s
1115	804	7.1756e+09	40	37	-	6.2500e+09	-	59.7	237s
1182	821	infeasible	73		-	6.2531e+09	-	62.7	240s
1245	832	6.8354e+09	23	30	-	6.2531e+09	-	67.6	246s
1256	838	6.7366e+09	29	64	-	6.2531e+09	-	68.9	284s
1266	841	6.7880e+09	34	23	-	6.2531e+09	-	94.1	288s
1272	850	6.7900e+09	36	18	-	6.2531e+09	-	94.1	294s
1287	860	6.8711e+09	39	12	-	6.2531e+09	-	97.9	296s
1342	912	8.8464e+09	50	22	-	6.2531e+09	-	102	304s
1400	915	infeasible	62		-	6.2675e+09	-	105	307s
1442	925	6.8032e+09	19	51	-	6.2676e+09	-	104	310s
1555	983	6.9731e+09	24	88	-	6.2676e+09	-	105	316s
1631	998	7.1914e+09	27	90	-	6.2676e+09	-	104	320s
1678	994	7.4123e+09	29	61	-	6.2676e+09	-	106	325s
1751	1043	7.0509e+09	37	95	-	6.2676e+09	-	109	333s
1791	1060	7.4551e+09	43	83	-	6.2676e+09	-	110	338s
1821	1089	8.5233e+09	47	18	-	6.2676e+09	-	111	343s
1882	1082	7.9693e+09	65	37	-	6.2774e+09	-	111	349s
1915	1109	6.3833e+09	17	118	-	6.2776e+09	-	117	353s
1959	1128	6.9336e+09	29	78	-	6.2777e+09	-	119	358s
2024	1161	6.3501e+09	18	216	-	6.2797e+09	-	120	363s
2081	1195	6.4456e+09	29	20	-	6.2797e+09	-	122	369s
2147	1168	8.7221e+09	41	122	-	6.2797e+09	-	127	377s
2160	1200	infeasible	44		-	6.2826e+09	-	131	384s
2198	1220	7.4779e+09	33	48	-	6.2831e+09	-	132	391s
2246	1244	infeasible	40		-	6.2861e+09	-	135	400s
2309	1324	6.7577e+09	24	38	-	6.2865e+09	-	135	407s
2412	1403	8.8706e+09	58	33	-	6.2944e+09	-	135	416s
2580	1416	7.4201e+09	50	34	-	6.2950e+09	-	131	424s
2667	1431	6.6756e+09	26	12	-	6.2962e+09	-	132	431s
2712	1543	6.6127e+09	18	49	-	6.3040e+09	-	133	441s

```
  2853  1595 7.2020e+09   37   22          - 6.3100e+09      -   133  449s
  2938  1616 9.2449e+09   52   30          - 6.3100e+09      -   137  458s
  2971  1632 1.0090e+10   57   31          - 6.3100e+09      -   141  466s
  3024  1691 infeasible   86               - 6.3221e+09      -   141  476s
  3128  1768 6.4237e+09   16   75          - 6.3273e+09      -   142  485s
  3237  1898 7.5197e+09   26   74          - 6.3313e+09      -   143  494s
  3393  2019 8.0203e+09   60   18          - 6.3359e+09      -   142  503s
  3578  2223 7.8604e+09   40   37          - 6.3369e+09      -   141  513s
  3840  2307 6.7115e+09   29    6          - 6.3419e+09      -   139  525s
  3960  2450 7.5401e+09   47    8          - 6.3434e+09      -   140  537s
H 4153  1475                    7.653903e+09 6.3462e+09  17.1%   140  548s
H 4241  1330                    7.437156e+09 6.3462e+09  14.7%   139  558s
H 4241   656                    6.820025e+09 6.3462e+09  6.95%   139  558s
  4452   654     cutoff   40      6.8200e+09 6.3563e+09  6.80%   137  568s
H 4616   496                    6.696134e+09 6.3778e+09  4.75%   134  568s
  4744   496     cutoff   19      6.6961e+09 6.3782e+09  4.75%   132  578s
  4967   525 infeasible   21      6.6961e+09 6.3974e+09  4.46%   131  588s
H 5200   343                    6.593085e+09 6.4154e+09  2.70%   129  598s
  5535   259 6.4608e+09   24  166 6.5931e+09 6.4358e+09  2.39%   126  606s
  5802   156     cutoff   24      6.5931e+09 6.4783e+09  1.74%   124  611s

Cutting planes:
  Cover: 91
  Implied bound: 1
  MIR: 24
  StrongCG: 20
  Flow cover: 174
  GUB cover: 12
  Zero half: 5
  Network: 12

Explored 6144 nodes (872292 simplex iterations) in 614.72 seconds (206.29 work units)
Thread count was 4 (of 4 available processors)

Solution count 5: 6.59309e+09 6.69613e+09 6.82003e+09 ... 7.6539e+09

Optimal solution found (tolerance 0.00e+00)
Best objective 6.593085384641e+09, best bound 6.593085384641e+09, gap 0.0000%
```

In [12]:

```
# print the objective value
print(m.objVal)
```

```
# retrieve the districts and their populations
#    but first get the district "centers"

centers = [ j for j in G.nodes if x[j,j].x > 0.5 ]

districts = [ [i for i in G.nodes if x[i,j].x > 0.5] for j in centers]
district_counties = [ [ G.nodes[i]["NAME20"] for i in districts[j] ] for j in range(k)]
district_populations = [ sum(G.nodes[i]["TOTPOP"] for i in districts[j]) for j in range(k) ]

# print district info
for j in range(k):
    print("District",j,"has population",district_populations[j],"and contains counties",district_counties[j])
    print("")
```

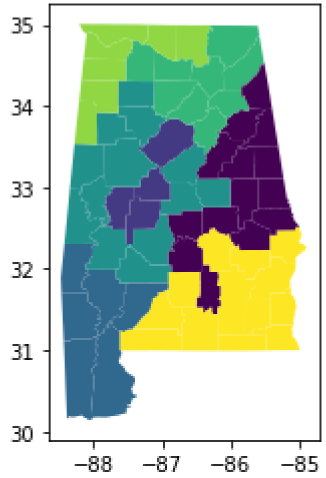
```
6593085384.640669
District 0 has population 714963 and contains counties ['Cleburne', 'Tallapoosa', 'Crenshaw', 'Calhoun', 'Elmore', 'Randolph',
'Clay', 'Cherokee', 'Lee', 'Chambers', 'Lowndes', 'Talladega', 'Autauga', 'Macon']

District 1 has population 720310 and contains counties ['Jefferson', 'Bibb', 'Hale', 'Perry']

District 2 has population 717488 and contains counties ['Clarke', 'Washington', 'Mobile', 'Monroe', 'Baldwin', 'Choctaw']

District 3 has population 718247 and contains counties ['Shelby', 'Dallas', 'Pickens', 'Walker', 'Chilton', 'Marengo', 'Greene',
'Fayette', 'Tuscaloosa', 'Wilcox', 'Coosa', 'Sumter', 'Winston']

District 4 has population 719832 and contains counties ['Lawrence', 'Morgan', 'Etowah', 'DeKalb', 'Marshall', 'Blount', 'Jackso
n', 'St. Clair', 'Cullman']

District 5 has population 717940 and contains counties ['Lauderdale', 'Madison', 'Lamar', 'Franklin', 'Colbert', 'Limestone', 'M
arion']

District 6 has population 715499 and contains counties ['Barbour', 'Geneva', 'Dale', 'Houston', 'Russell', 'Conecuh', 'Montgomer
y', 'Coffee', 'Covington', 'Henry', 'Butler', 'Escambia', 'Pike', 'Bullock']
```

In [13]:
```
# Let's draw it on a map
import geopandas as gpd
```

In [14]:
```
# Read Alabama county shapefile from "AL_county.shp"
filepath = 'C:\\Users\\blrod\\Downloads\\districting-data-2020-county\\'
filename = 'AL_county.shp'
```

```
# Read geopandas dataframe from file
df = gpd.read_file( filepath + filename )
```

In [15]:
```
# Which district is each county assigned to?
assignment = [ -1 for i in G.nodes ]

labeling  = { i : -1 for i in G.nodes }
for j in range(k):
    district = districts[j]
    for i in district:
        labeling[i] = j

# Now add the assignments to a column of the dataframe and map it
node_with_this_geoid = {G.nodes[i]['GEOID20'] : i for i in G.nodes}

#pick a position u in the dataframe
for u in range(G.number_of_nodes()):

    geoid = df['GEOID20'][u]

    # what node in G has thus geoid?
    i = node_with_this_geoid[geoid]

    # position u in the dataframe should be given
    # the same district # that county i has in 'labeling'
    assignment[u] = labeling[i]

#now add the assignments to a column of our dataframe and then map it
df['assignment'] = assignment

my_fig = df.plot(column='assignment').get_figure()
```




```
In [ ]:
```