

Arkansas(ar) redistricting report

Dr. Hamidreza Validi, IE5318



December 1, 2023

imse|TTU

Ali Azizi Deh Sorkh

Contents

[Executive Summary Letter 2](#_Toc153183390)

[Introduction 2](#_Toc153183391)

[Criteria 2](#_Toc153183392)

[Problem Statement 3](#_Toc153183393)

[Problem Statement and OR model (words, math) 3](#_Toc153183394)

[Python/Gurobi Code 6](#_Toc153183395)

[Min Cut edge 6](#_Toc153183396)

[Min Moment of Inertia 12](#_Toc153183397)

[Min Perimeter 19](#_Toc153183398)

[System Specifications 25](#_Toc153183399)

[Maps and pictures 25](#_Toc153183400)

[Min Perimeter and Min Cut edge 25](#_Toc153183401)

[Min Moment of Inertia and 26](#_Toc153183402)

[Evaluation Plans 27](#_Toc153183403)

[Conclusion 27](#_Toc153183404)

# Executive Summary Letter

Political redistricting holds immense significance in the United States due to the nation's deep-rooted commitment to democratic ideals. Ensuring equitable design of political districts is paramount as it safeguards the integrity of Americans' votes and fosters a more balanced society. This report will elaborate on our approach to addressing the challenge of political redistricting in the state of Arkansas.

Initially, we examined both federal and state redistricting criteria, followed by gathering data to support our ongoing development. Subsequently, a linear program was devised to construct a model showcasing a potential districting layout for Arkansas.

The devised redistricting blueprint revolves around establishing districts with an approximate populace of 750,000 individuals per district. This plan maintains a population deviation of 0.28%, aligning faithfully with the redistricting standards mandated at both state and federal levels.

# Introduction

Every decade, as mandated by the United States Constitution, a census is conducted. The outcomes of this survey dictate whether federal and state governing bodies need to undergo redistricting, reassignment, or redistribution.

Redistricting refers to the process of dividing a political entity into smaller, nearly equal population segments to prevent favoritism toward specific demographics or regions. Any proposed district scheme must adhere to the redistricting criteria established by both federal and state authorities.

Given variations in population across diverse spatial areas and the constraints imposed by geographic, political, and social factors, redistricting poses a highly intricate challenge often resolved through optimization models. The approach employed in this solution utilizes the minimum cut edges technique.

# Criteria

**Federal Criteria**

Every state within the United States of America must abide by the following criteria points:

● Each district must have a nearly equal population in relation to one another.

○ Apportionment Clause, Article, Section 2, U.S. Constitution

● Each district must not intentionally discriminate racially.

○ Voting Rights Act, 1965

These points are enforced at the federal level. In addition to this, each state has the freedom to

impose their criteria.

**State of Arkansas Criteria**

The state of Arkansas requires that the following criteria points are followed:

● Each district must maintain a +/- 5% population deviation.

● Each district must not intentionally discriminate or favor any race.

● Districts cannot be redrawn strictly based on race.

Source: https://arkansasredistricting.org/about-the-process/redistricting-criteria-and-goals/

In the redistricting process in Arkansas, there are few specific constraints imposed. Ideally, districts should be compact and connected, while also maintaining the integrity of counties, political divisions, communities of shared interests, and existing districts. Whenever feasible, efforts are made to avoid grouping incumbents within the same district. Factors considered in selecting a districting plan involve assessing its impact on incumbents, its use of partisan information, and its overall competitiveness.

## Problem Statement

We'll employ operations research methodology to craft a districting blueprint for Arkansas that complies with both federal and state governmental constraints.

## Problem Statement and OR model (words, math)

The models for Processes 1 and 2 exhibit considerable similarity. Hence, we will solely focus on the model that generated the most optimal redistricting map. The objective of this process is to minimize the total length of cut edges, ensuring compactness for each district. Preserving counties in their entirety is a priority during this process.

I'm unable to directly manipulate external files or formats, but I can provide the information in a format that you can copy and paste into a Word document:

Sets:

- (C) represents the set of counties (Nodes) 1, 2, ..., 74

- (J) represents the set of districts 1, 2, ..., (k)

- (E) represents the set of edges 1, 2, ..., (e)

- (N(i)) is the set of neighbors of county (i)

Indices:

- (i) denotes a county

- (j) denotes a district

- (u) and (v) are counties that are being checked for continuity

Parameters:

- (P\_i) = population of county (i)

- (totalpop\_j) = total population of district (j)

- (e) = total number of edges

- (n) = number of counties (nodes)

- (k) = total number of desired districts

- (L) = lower bound maximum deviation

- (U) = upper bound maximum deviation

- (M) = number of counties - number of districts + 1

Variables:

- (x{ij}) = {1 if county (i) is a part of district (j), 0 otherwise}

- (y\_{uv}) = {1 if there is a district boundary between counties (u) and (v), 0 otherwise}

- (r\_{ij}) = {1 if county (i) is the root of district (j), 0 otherwise}

- (f\_{ij}) = {1 if flow is being sent from (i) to (j), 0 otherwise}

This information can be copied and pasted into a Word document or any text editor of your choice.

A white sheet of paper with black text

Description automatically generated

# Python/Gurobi Code

I have run three different models based on Dr. Buchanan's provided information on this link: <https://github.com/AustinLBuchanan/Districting-Examples-2020>

## Min Cut edge

**The code**: (All codes are available at my GitHub Directory )

#import all necessary packages

import gurobipy as gp

from gurobipy import GRB

from gerrychain import Graph

import networkx as nx

import geopandas as gpd

import math

#Set filepath and filename equal to the path/name of the data used respectively

filepath = r'C:\Users\aliaz\Downloads\IEM40132020RedistrictingProject-main\IEM40132020RedistrictingProject-main/'

filename= 'AR\_county.json'

#Create a new Graph object G from the file

G = Graph.from\_json(filepath + filename)

#Set each node in G to be equal to the population of their respective county

for node in G.nodes:

G.nodes[node]['TOTPOP'] = G.nodes[node]['P0010001']

#Print each node, the county it represents, and their 2020 population

for node in G.nodes:

name = G.nodes[node]['NAME20']

population = G.nodes[node]['TOTPOP']

print("Node",node,"represents",name,"County with 2020 population of",population)

#draw the graph of nodes

nx.draw(G, with\_labels=True)

#set the ceiling and floor of the model equal to the maximum deviation/2 \* the average population

dev = 0.01

k = 4

tot\_pop = sum(G.nodes[node]['TOTPOP'] for node in G.nodes)

L = math.ceil((1-dev/2)\*tot\_pop/k)

U = math.floor((1+dev/2)\*tot\_pop/k)

print("Using L =",L,"and U =",U,"and k =",k)

#create a new model object and create variables

m = gp.Model()

x = m.addVars(G.nodes, k, vtype=GRB.BINARY)

y = m.addVars(G.edges, vtype=GRB.BINARY)

#set objective to minimize cut edges

m.setObjective( gp.quicksum( y[u,v] for u,v in G.edges ), GRB.MINIMIZE )

# each county i is assigned to a district j

m.addConstrs(gp.quicksum(x[i,j] for j in range(k)) == 1 for i in G.nodes)

# each district j has a population at least L and at most U

m.addConstrs( gp.quicksum( G.nodes[i]['TOTPOP'] \* x[i,j] for i in G.nodes) >= L for j in range(k))

m.addConstrs( gp.quicksum( G.nodes[i]['TOTPOP'] \* x[i,j] for i in G.nodes) <= U for j in range(k))

# an edge is cut if u is assigned to district j but v is not.

m.addConstrs( x[u,j] - x[v,j] <= y[u,v] for u,v in G.edges for j in range(k))

m.update()

# add root variables: r[i,j] equals 1 if node i is the root of district j

r = m.addVars( G.nodes, k, vtype=GRB.BINARY)

import networkx as nx

DG = nx.DiGraph(G)

f = m.addVars(DG.edges)

# The big-M proposed by Hojny et al.

M = G.number\_of\_nodes() - k + 1

# each district should have one root

m.addConstrs( gp.quicksum( r[i,j] for i in G.nodes ) == 1 for j in range(k) )

# If node i isn't assigned to district j, then it cannot be its root

m.addConstrs( r[i,j] <= x[i,j] for i in G.nodes for j in range(k) )

# If not a root, consume some flow

# If a root, only send out (so much) flow

m.addConstrs( gp.quicksum( f[j,i] - f[i,j] for j in G.neighbors(i) )

>= 1 - M \* gp.quicksum( r[i,j] for j in range(k) ) for i in G.nodes )

# Do not send flow across cut edges

m.addConstrs( f[i,j] + f[j,i] <= M \* (1-y[i,j] )for i,j in G.edges)

m.update()

# sole IP model

m.optimize()

print("The number of cut edges is",m.objval)

# retrieve the districts and their population

districts = [[i for i in G.nodes if x[i,j].x > 0.5] for j in range(k)]

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 it

for j in range(k):

print("District",j,"has population",district\_populations[j],"and contains counties",district\_counties[j])

print("")

# Read Arkansas county shapefile from "AR\_county.shp"

filepath = r'C:\Users\aliaz\Downloads\IEM40132020RedistrictingProject-main\IEM40132020RedistrictingProject-main/'

filename = 'AR\_county.shp'

# Read geopandas dataframe from file

df = gpd.read\_file( filepath + filename)

# Which district is each county assigned to?

assignment = [ -1 for i in G.nodes ]

labeling = { i : j for i in G.nodes for j in range(k) if x[i,j].x > 0.5 }

# add 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 data frame

for u in range(G.number\_of\_nodes()):

geoid = df['GEOID20'][u]

i = node\_with\_this\_geoid[geoid]

assignment[u] = labeling[i]

#print the map

df['assignment'] = assignment

my\_fig = df.plot(column='assignment').get\_figure()

**Result**:

1. Node Graph

A network of blue dots and lines

Description automatically generated

1. Optimization

A black and white image of a number

Description automatically generated with medium confidence

The number of cut edges is 33.0.

District 0 has a population of 751754 and contains counties ['Franklin', 'Crawford', 'Benton', 'Madison', 'Sebastian', 'Washington']

District 1 has population 754547 and contains counties ['Little River', 'Ashley', 'Desha', 'Montgomery', 'Howard', 'Nevada', 'Grant', 'Dallas', 'Cleveland', 'Lafayette', 'Chicot', 'Pope', 'Bradley', 'Drew', 'Pike', 'Union', 'Hempstead', 'Polk', 'Clark', 'Logan', 'Miller', 'Arkansas', 'Johnson', 'Garland', 'Sevier', 'Jefferson', 'Lincoln', 'Scott', 'Hot Spring', 'Columbia', 'Ouachita', 'Yell', 'Calhoun', 'Phillips']

District 2 has a population of 750788 and contains counties ['Faulkner', 'Conway', 'Pulaski', 'Saline', 'Lonoke', 'Perry']

District 3 has a population of 754435 and contains counties ['Jackson', 'Clay', 'Baxter', 'Boone', 'St. Francis', 'Sharp', 'Greene', 'Woodruff', 'White', 'Lee', 'Crittenden', 'Marion', 'Prairie', 'Lawrence', 'Poinsett', 'Stone', 'Independence', 'Fulton', 'Carroll', 'Van Buren', 'Searcy', 'Randolph', 'Izard', 'Craighead', 'Cleburne', 'Monroe', 'Mississippi', 'Newton', 'Cross']

1. Map

A map of arkansas with different colored states

Description automatically generated

## Min Moment of Inertia

**The code**:

from gerrychain import Graph

import gurobipy as gp

from gurobipy import GRB

import math

from geopy.distance import geodesic

import geopandas as gpd

#Set filepath and filename equal to the path/name of the data used respectively

filepath = r'C:\Users\aliaz\OneDrive\Desktop\Progress\Courses\Fall2023\OR\Final Project/'

filename= 'AR\_county.json'

#Create a new Graph object G from the file

G = Graph.from\_json(filepath + filename)

for node in G.nodes:

G.nodes[node]['TOTPOP'] = G.nodes[node]['P0010001']

G.nodes[node]['C\_X'] = G.nodes[node]['INTPTLON20']

G.nodes[node]['C\_Y'] = G.nodes[node]['INTPTLAT20']

# 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

dev = 0.01

k = 4

tot\_pop = sum(G.nodes[node]['TOTPOP'] for node in G.nodes)

L = math.ceil((1-dev/2)\*tot\_pop/k)

U = math.floor((1+dev/2)\*tot\_pop/k)

print("Using L =",L,"and U =",U,"and k =",k)

m = gp.Model()

x = m.addVars(G.nodes, G.nodes, vtype=GRB.BINARY)

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.MINIMIZE )

# 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()

# add contiguity constraints

import networkx as nx

DG = nx.DiGraph(G)

# add flow variables

# f[i,j,v] = flow across arc (i,j) that is sent from source/root v

f = m.addVars( DG.edges, G.nodes )

# 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 receive 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 node j cannot receive 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 )

m.update()

m.Params.MIPGap = 0.0

m.optimize()

# 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("")

filepath = 'C:/Users/Logan/Desktop\College/IEM4013Project/'

filename = 'AR\_county.shp'

df = gpd.read\_file( filepath + filename)

# 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 this 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()

**Results**

District 0 has population 753326 and contains counties ['Jackson', 'Clay', 'Baxter', 'St. Francis', 'Sharp', 'Greene', 'Woodruff', 'White', 'Lee', 'Crittenden', 'Marion', 'Prairie', 'Lawrence', 'Poinsett', 'Independence', 'Lonoke', 'Fulton', 'Arkansas', 'Randolph', 'Izard', 'Craighead', 'Cleburne', 'Monroe', 'Mississippi', 'Cross', 'Phillips']

District 1 has population 749461 and contains counties ['Franklin', 'Faulkner', 'Boone', 'Conway', 'Pulaski', 'Madison', 'Pope', 'Stone', 'Van Buren', 'Johnson', 'Searcy', 'Perry']

District 2 has population 755116 and contains counties ['Little River', 'Ashley', 'Desha', 'Montgomery', 'Howard', 'Nevada', 'Grant', 'Dallas', 'Cleveland', 'Lafayette', 'Saline', 'Chicot', 'Bradley', 'Drew', 'Pike', 'Union', 'Hempstead', 'Polk', 'Clark', 'Logan', 'Miller', 'Garland', 'Sevier', 'Jefferson', 'Lincoln', 'Scott', 'Hot Spring', 'Columbia', 'Ouachita', 'Yell', 'Calhoun']

District 3 has population 753621 and contains counties ['Crawford', 'Benton', 'Sebastian', 'Carroll', 'Washington', 'Newton']

**A close up of a screen

Description automatically generated**

**Map**

**A map of arkansas with different colored states

Description automatically generated**

## Min Perimeter

**The code:**

from gerrychain import Graph

filepath = r'C:\Users\aliaz\OneDrive\Desktop\Progress\Courses\Fall2023\OR\Final Project/'

filename= 'AR\_county.json'

G = Graph.from\_json(filepath + filename)

for node in G.nodes:

G.nodes[node]['TOTPOP'] = G.nodes[node]['P0010001']

dev = 0.01

import math

k = 4

tot\_pop = sum(G.nodes[node]['TOTPOP'] for node in G.nodes)

L = math.ceil((1-dev/2)\*tot\_pop/k)

U = math.floor((1+dev/2)\*tot\_pop/k)

print("Using L =",L,"and U =",U,"and k =",k)

import gurobipy as gp

from gurobipy import GRB

# create model

m = gp.Model()

# create variables

x = m.addVars(G.nodes, k, vtype=GRB.BINARY) # x[i,j] equals one when county i is assigned to district j

y = m.addVars(G.edges, vtype=GRB.BINARY) # y[u,v] equals one when edge {u,v} is cut

m.setObjective( gp.quicksum( G.edges[u,v]['shared\_perim'] \* y[u,v] for u,v in G.edges ), GRB.MINIMIZE )

m.addConstrs( gp.quicksum( x[i,j] for j in range(k)) == 1 for i in G.nodes )

# add constraints saying that each district has population at least L and at most U

m.addConstrs( gp.quicksum( G.nodes[i]['TOTPOP'] \* x[i,j] for i in G.nodes) >= L for j in range(k) )

m.addConstrs( gp.quicksum( G.nodes[i]['TOTPOP'] \* x[i,j] for i in G.nodes) <= U for j in range(k) )

# add constraints saying that edge {u,v} is cut if u is assigned to district j but v is not.

m.addConstrs( x[u,j] - x[v,j] <= y[u,v] for u,v in G.edges for j in range(k) )

m.update()

# Add root variables: r[i,j] equals 1 if node i is the "root" of district j

r = m.addVars( G.nodes, k, vtype=GRB.BINARY)

# To solve the MIP faster, fix some district roots:

r[20,0].LB = 1 # fix Oklahoma county as root of district 0

r[37,1].LB = 1 # fix Tulsa county as root of district 1

r[62,2].LB = 1 # fix Comanche county as root of district 2

r[56,3].LB = 1

# Add flow variables: f[u,v] = amount of flow sent across arc uv

# Flows are sent across arcs of the directed version of G which we call DG

import networkx as nx

DG = nx.DiGraph(G) # directed version of G

f = m.addVars( DG.edges )

# The big-M proposed by Hojny et al.

M = G.number\_of\_nodes() - k + 1

# Each district j should have one root

m.addConstrs( gp.quicksum( r[i,j] for i in G.nodes ) == 1 for j in range(k) )

# If node i is not assigned to district j, then it cannot be its root

m.addConstrs( r[i,j] <= x[i,j] for i in G.nodes for j in range(k) )

# if not a root, consume some flow.

# if a root, only send out (so much) flow.

m.addConstrs( gp.quicksum( f[j,i] - f[i,j] for j in G.neighbors(i) )

>= 1 - M \* gp.quicksum( r[i,j] for j in range(k) ) for i in G.nodes )

# do not send flow across cut edges

m.addConstrs( f[i,j] + f[j,i] <= M \* ( 1 - y[i,j] ) for i,j in G.edges )

m.update()

m.optimize()

print("The number of cut edges is", m.ObjVal)

districts = [[i for i in G.nodes if x[i,j].x > 0.5] for j in range(k)]

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)]

for j in range(k):

print("District",j,"has population",district\_populations[j],"and contains counties",district\_counties[j])

print("")

import geopandas as gpd

filepath = r'C:\Users\aliaz\OneDrive\Desktop\Progress\Courses\Fall2023\OR\Final Project/'

filename = 'AR\_county.shp'

df = gpd.read\_file( filepath + filename)

assignment = [ -1 for i in G.nodes ]

labeling = { i : j for i in G.nodes for j in range(k) if x[i,j].x > 0.5 }

node\_with\_this\_geoid = { G.nodes[i]['GEOID20'] : i for i in G.nodes }

for u in range(G.number\_of\_nodes()):

geoid = df['GEOID20'][u]

i = node\_with\_this\_geoid[geoid]

assignment[u] = labeling[i]

df['assignment'] = assignment

my\_fig = df.plot(column='assignment').get\_figure()

**Result:**

The number of cut edges is 12.00964554654077.

District 0 has population 750788 and contains counties ['Faulkner', 'Conway', 'Pulaski', 'Saline', 'Lonoke', 'Perry']

District 1 has population 751754 and contains counties ['Franklin', 'Crawford', 'Benton', 'Madison', 'Sebastian', 'Washington']

District 2 has population 754435 and contains counties ['Jackson', 'Clay', 'Baxter', 'Boone', 'St. Francis', 'Sharp', 'Greene', 'Woodruff', 'White', 'Lee', 'Crittenden', 'Marion', 'Prairie', 'Lawrence', 'Poinsett', 'Stone', 'Independence', 'Fulton', 'Carroll', 'Van Buren', 'Searcy', 'Randolph', 'Izard', 'Craighead', 'Cleburne', 'Monroe', 'Mississippi', 'Newton', 'Cross']

District 3 has population 754547 and contains counties ['Little River', 'Ashley', 'Desha', 'Montgomery', 'Howard', 'Nevada', 'Grant', 'Dallas', 'Cleveland', 'Lafayette', 'Chicot', 'Pope', 'Bradley', 'Drew', 'Pike', 'Union', 'Hempstead', 'Polk', 'Clark', 'Logan', 'Miller', 'Arkansas', 'Johnson', 'Garland', 'Sevier', 'Jefferson', 'Lincoln', 'Scott', 'Hot Spring', 'Columbia', 'Ouachita', 'Yell', 'Calhoun', 'Phillips']

**A screenshot of a computer

Description automatically generated**

**Map:**

**A map of arkansas with different colored areas

Description automatically generated**

# System Specifications

All models have been run on this system:

A black text on a white background

Description automatically generated

# Maps and pictures

## Min Perimeter and Min Cut edge

<https://districtr.org/plan/214548>

A map of arkansas with different colored squares

Description automatically generated

## Min Moment of Inertia and

<https://districtr.org/plan/126890>

A map of the state of arkansas

Description automatically generated

# Evaluation Plans

Two new district plans for Arkansas were created. The first plan employed minimum cutting edges, resulting in a maximum population deviation of 0.28%. Process two utilized the minimum moment of inertia, determining a maximum population deviation of 0.45%. Process one is recommended due to its compliance with all federal and state criteria.

# Conclusion

Upon reviewing federal and state redistricting criteria, data was gathered to develop a linear program depicting potential districting plans for the state of Arkansas. The criteria outlined in the proposed districting plan were covered. Each district was determined to encompass a population of 750,000 people. The population for each district, along with the counties they cover, is as follows:

- District 1: Population of 751,754, including counties such as Franklin, Crawford, Benton, Madison, Sebastian, and Washington.

- District 2: Population of 754,435, covering counties including Jackson, Clay, Baxter, Boone, St. Francis, Sharp, Greene, Woodruff, White, Lee, Crittenden, Marion, Prairie, Lawrence, Poinsett, Stone, Independence, Fulton, Carroll, Van Buren, Searcy, Randolph, Izard, Craighead, Cleburne, Monroe, Mississippi, Newton, and Cross.

- District 3: Population of 750,788, encompassing counties like Faulkner, Conway, Pulaski, Saline, Lonoke, and Perry.

- District 4: Population of 754,547, containing counties such as Little River, Ashley, Desha, Montgomery, Howard, Nevada, Grant, Dallas, Cleveland, Lafayette, Chicot, Pope, Bradley, Drew, Pike, Union, Hempstead, Polk, Clark, Logan, Miller, Arkansas, Johnson, Garland, Sevier, Jefferson, Lincoln, Scott, Hot Spring, Columbia, Ouachita, Yell, Calhoun, and Phillips.

The mapped districts maintain contiguity following the guidelines. The maximum population deviation of 0.28% from the minimum cut-edge method indicates that the population is nearly equivalent across districts.

# References

Github: <https://github.com/DrsAz77/Districting-OR.git>

Codes: <https://github.com/logandavis2518/IEM4013_2020RedistrictingProject.git>

Data Source: <https://github.com/AustinLBuchanan/Districting-Examples-2020>

Redistricting Sources:

<https://arkansasredistricting.org/about-the-process/redistricting-criteria-and-goals/>

Voting Rights Act, 1965

Apportionment Clause, Article, Section 2, U.S. Constitution