Voting District Generation

Be sure to sign up for “Volunteer” or in-class project presentations

Project Teams

- Teams listed on course home page
- All class members are now part of a team

3-student teams will have a use case number target appropriate to the size of the team
Comments on Project Use Case List

- Project not a great fit for use cases since actor driven scenarios create many complex use cases.
- List of use cases is a 2-step process:
  - Teams develop their list of requested use cases.
  - Recommendations of each team will be considered in the generation of a master list of about 40 use cases.
- Use cases for project will:
  - Provide balanced units of work.
  - Allow for final demo grading based on completed use cases.
- Use cases will consist of required, preferred, and optional.
- Project grading will emphasize required and preferred use cases.

Project - Voting District Generation

- Develop a system to generate a state congressional redistricting plan based on user-set parameters.
- Parameters include, but are not limited to:
  - Compactness
  - Alignment with county and city boundaries
  - Variation limits in population
  - Alignment with natural boundaries (e.g., highways, rivers, etc.)
  - Adherence to Voting Rights Act
- Shows a comparison with existing district boundaries.
- Depicts the results graphically in a Web interface.
- Works for a minimum of 3 states.
**Top Level View of the System**

- Consider a state with a defined set of geographically defined districts (each small in size). We refer to these as precincts.
- A higher level district (e.g., congressional district) is composed of some collection of these small districts.
- Determine a suitable allocation of low-level districts to higher level districts.
- The process:
  - ignores political considerations
  - is "good" in some sense
  - Does not violate any constraints

---

**Algorithmic Approach – Part 1**

- Create an initial set of congressional districts by
  - In a given step, combine some (not all) pairs of precincts according to the factors set by the user – producing a set of precinct clusters.
  - Repeat until the number of precinct clusters equals the desired number of congressional districts.

*Rules for combining clusters might change as the algorithm proceeds.*
Algorithmic Approach - Part 2

Starting with the clusters obtained in Part 1

1. Move one precinct into a different higher level district
2. Determine if the move violates any constraints, and if so, move that district back
3. Measure the “goodness” of each district affected by the move. If the move improves overall goodness, keep the move.
4. Repeat steps 2-4 until goodness cannot be further improved

This approach is referred to as “simulated annealing”

If you did not cover this algorithmic approach in CSE373, be sure to read the relevant chapter in the Skiena book

What are Some of the Constraints?

- Districts should be contiguous, where possible
- Incumbent representatives should be in separate districts, where possible
- Districts should be aligned on natural boundaries (e.g., rivers and highways) where appropriate
- Districts should not split counties and/or cities
- Majority/minority districts should be allowed (where appropriate)
- Communities of interest should be enclosed in a district (where possible)
- Districts should be compact
What are Some of the “Goodness” Measures?

- Compactness
- Contiguity
- Equal population
- Preservation of existing communities
- Partisan fairness
- Majority/minority

These measures will have weights in your “goodness” calculation

Some measures are quantifiable, other perhaps not

Compactness Measures

- Not a single best measure
- Good description of the measures in Kaufman, King, and Komisarchik
- You should plan to use multiple compactness measures in your objective function
- Think about using a simple graph based compactness measure for in-progress algorithm calculations, with more precise measures for GUI

Compactness code skeleton will be given to you
Graph Theoretical Compactness

- Not well recognized, but possible very effective
- Dube paper - link in class web site references
- Represent precincts as nodes in a graph with edges connecting nodes if the precincts are adjacent
- Requires preprocessing your data so that you construct the graph
- Minimize the number of edge cuts between districts

Partisan Fairness Measures

- You should include multiple measures of partisan fairness
  - Efficiency gap - required
  - Seats - to - votes curve analysis
  - Mean-median difference
  - Others

Some measures will be extracted from previous projects, and provide as an API.

These measures estimate the presence of political gerrymandering
Equal Population Measure

- Higher level districts (e.g., congressional districts) should contain approximately equal populations
- No uniform definition of equal population
- Consider an approach in which you:
  1. Calculate the ideal population per district (state population / # districts)
  2. Calculate the deviation of a district from the ideal population
  3. Do a look-up on a curve to calculate the equal population measure

Majority-Minority Districts

- Your system should have an option to have your state districting include majority-minority districts
- Requirement that some states include some number of districts where the district voters are more than 50% in a stated minority
- Related laws sometimes used to pack districts
- Your system will include a user option to
  1. Define # of majority-minority districts
  2. Define min/max thresholds (e.g., 50% / 60%)
  3. Define the ethnic/racial groups considered

Be sure to read the Wikipedia page referenced on the CSE308 home page
What Data is Needed for the Constraints and Measures?

- Geospatial boundary data
  - Precincts
  - Existing Congressional districts
  - Cities/counties
  - Natural boundaries (e.g., roads and rivers)
  - Census tracts (including demographic data)
- Election results data
- Voter preference data (e.g., party affiliation)

GUI

- Your user interface will be very helpful in understanding system requirements
- Good to begin work on the GUI very soon.

Demo of a previous system in class
Preprocessing

- You will need to process the raw data so that you have data structures that can support efficient run-time operation.

- Some tasks:
  - Build a data structure that includes nodes (i.e., precincts) and edges (neighboring precincts).
  - Decompose geometric files into data for individual precincts.
  - Determine the set of neighboring precincts for each precinct (requires geospatial analysis).
  - Map census data (e.g., demographics) to nodes.

Web Site Project Page

- CSE308 Web site now includes the allocation of project teams to states.
- Only states with more than 1 congressional representative are included.
- Send me an e-mail with your team’s choice of states. Good to send 4-5 requests in priority order.
- Good to select a big state, a small (easy) state, and an interesting state.
- You can change states any time during the semester (e.g., if data not available).
Use Case List Grading Process

- You develop your initial list of use cases.
- After requirements document is submitted, you will be given a master list of project use case.
- You modify your list – and build.
- For the final demo, you bring a list of your use cases mapped to the master list.

Text Use Case Example

- Sample file (last semester’s list of use cases) contained in CSE308 Web site.
In-class Oral Communications Points

- TAs available in class starting next Monday
- Points for volunteer presentations
- Rosters for this semester will be posted once we have some grades in the roster (probably next week)
- Additional in-class oral communications points for significant contributions to the class

Review Status of High Priority Project Tasks

- Understand terminology and concepts in problem domain (read background references)
- Select 3 states to use in your development DB
- Read the districting language in the constitutions of your 3 states
- Understand advantages and limitations of algorithmic options
- Think about the components in your GUI
- Think about data normalization (how do you use different metrics in a single objective function)
- Start to write your objective function (on paper)
- Think about your system approach to grouping scenarios
- Build a simple system prototype to help understand SW design issues