Automated Redistricting Simulation Using GFlowNets

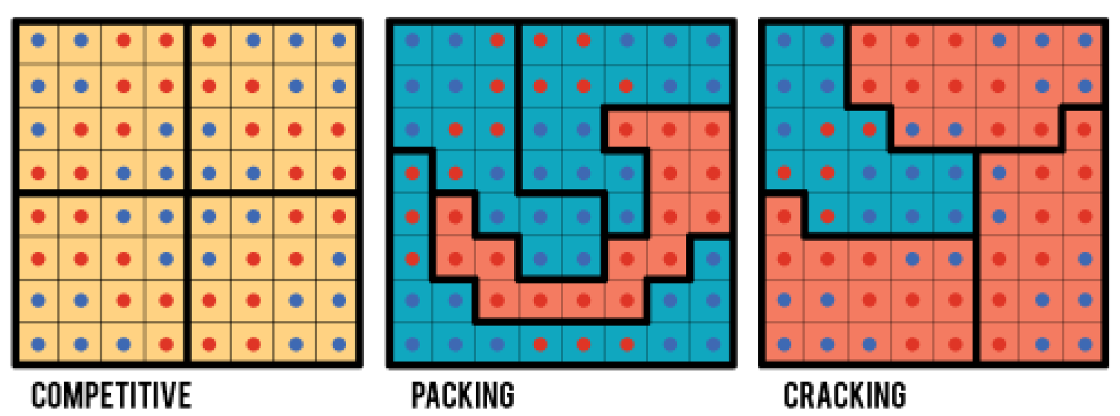
Arnaud Bergeron, Kevin Lessard and Alex Maggioni (alphabetically)

Mila Québec & Université de Montréal

Problem Setting

Gerrymandering is the practice of drawing electoral district boundaries to unfairly favor one political party or group. This manipulation is achieved primarily through two techniques:

- **Packing:** Concentrating a particular group of voters into a few districts, reducing their influence elsewhere.
- **Cracking:** Splitting a group of voters across multiple districts to dilute their influence.



These tactics distort democratic processes, emphasizing the need for transparent and fair redistricting algorithms.

Our Contribution

We replicate the results of the *Markov chain Monte Carlo* (MCMC) algorithms presented in the paper *Automated Redistricting Simulation Using Markov Chain Monte Carlo*. The MCMC algorithm incorporates contiguity and equal population constraints to produce valid districting plans.

To extend these results, we implement a novel *GFlowNet* approach to redistricting. GFlowNets allow for efficient exploration of diverse redistricting solutions, complementing the MCMC framework by improving the sampling process.

We use data from the **ALARM Project**, an open-source initiative providing comprehensive redistricting data for the United States.

Important Metrics

Compactness: Compact districts minimize irregular shapes, promoting fairness. We measure compactness using the *Polsby-Popper Score*

Population Constraint: Districts must have nearly equal populations. We evaluate this constraint using the entropy of the population distribution across districts, where higher entropy indicates a more even population balance.

Partisan Bias: Measures whether one party gains an unfair advantage due to redistricting. It answers the question: How would the outcome look in a perfectly even 50/50 statewide election?

Efficiency Gap: Quantifies wasted votes not contributing to a win to identify unfair advantages and detect potential packing and cracking strategies.

Example

Consider a state with 3 districts and 300 total votes. The statewide vote share is 60% Democrat and 40% Republican with 100 votes per district.

District	Dem.	Rep.	Adj. Winner
1	80 → **70**	20 → **30**	Democrat
2	80 → **70**	20 → **30**	Democrat
3	20 → **10**	80 → **90**	Republican
Adj. Total	150	150	Dem. :)

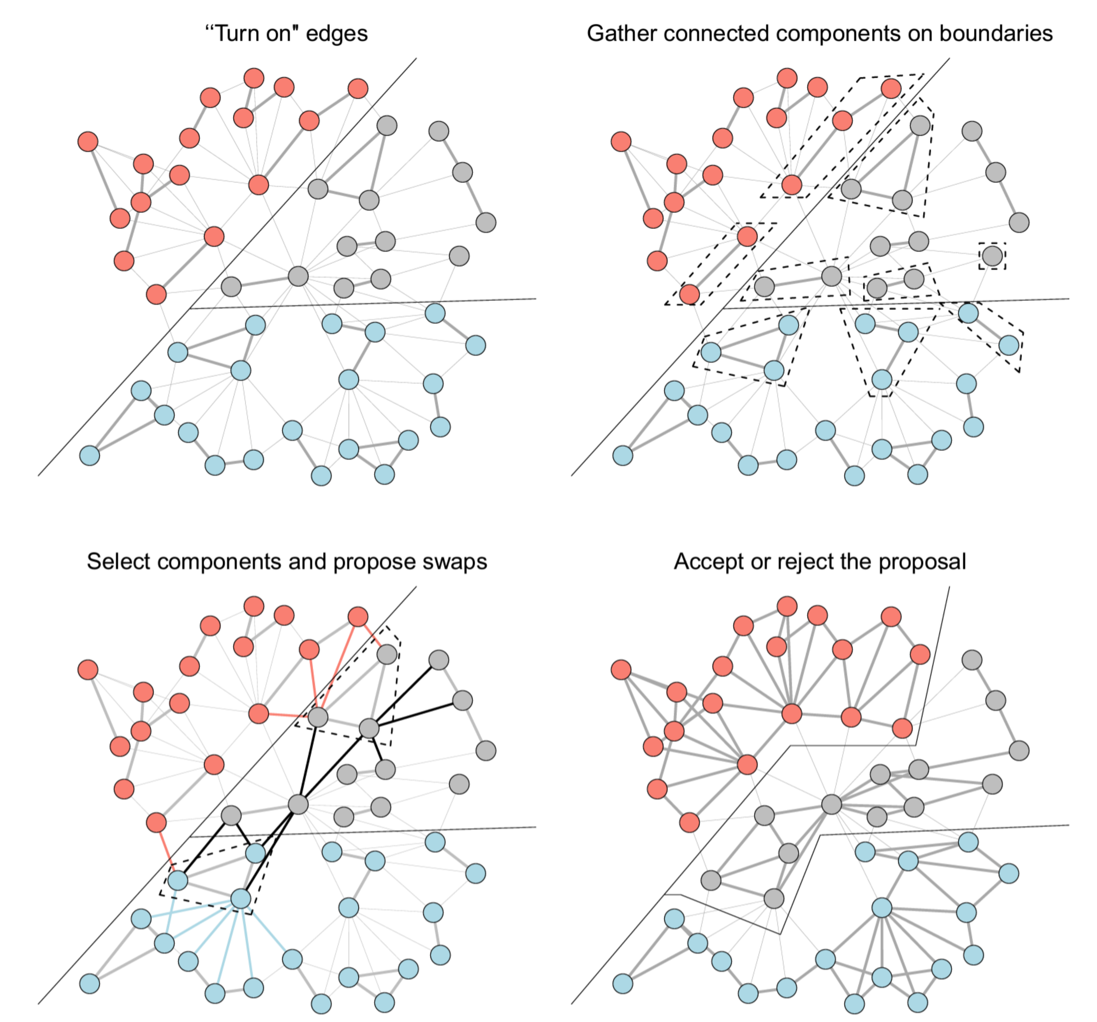
Table 1:Partisan Gap Example

District	Dem.	Rep.	Winner
1	51 + 29	20	Democrat
2	51 + 29	20	Democrat
3	20	51 + 29	Republican
Wasted votes	78	69	Dem. :(

Table 2:Efficiency Gap Example

Core MCMC Algorithm

The following steps form the foundation of the Markov Chain Monte Carlo (MCMC) redistricting algorithms:

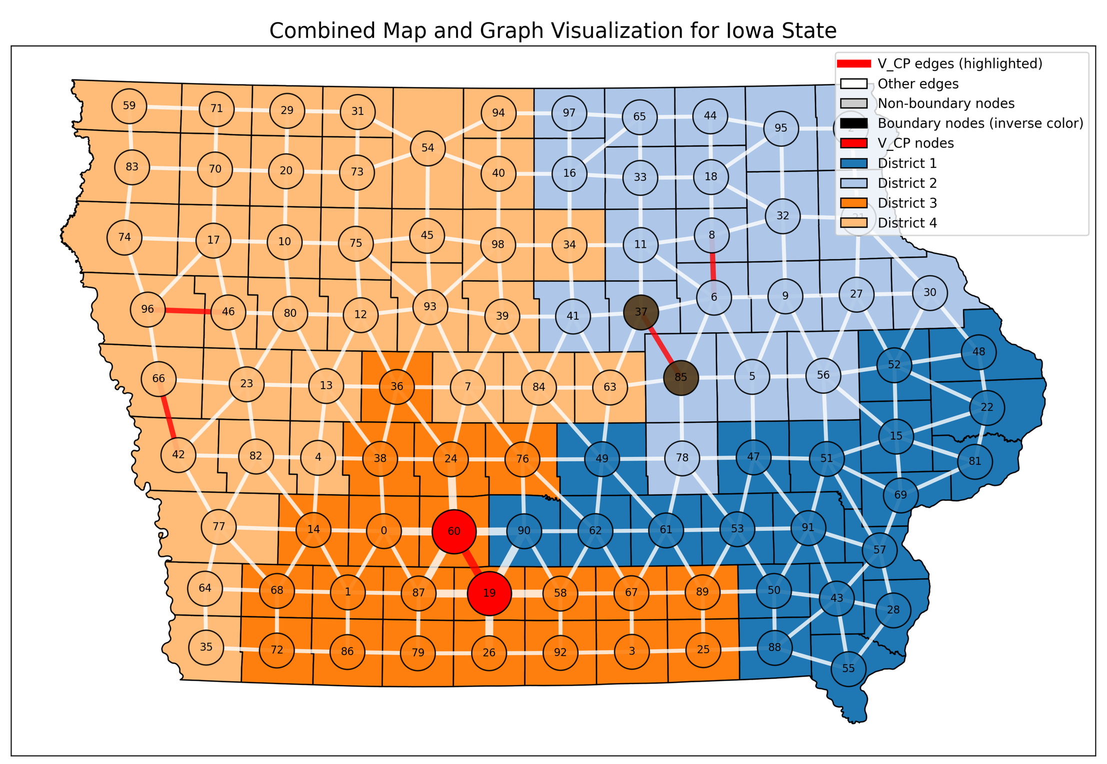


1. **Initialization**: Start from a valid partition of the graph into contiguous districts.
2. **Turn On Edges**: Randomly activate edges between nodes (precincts) with a small probability q and gather connected components.
3. **Boundary Identification**: Identify all connected components along the boundaries of districts using BFS.
4. **Select Components for Swapping**: Randomly choose a subset of non-adjacent connected components along the boundaries using the Zero-truncated Poisson distribution.
5. **Propose Swaps**: Reassign the chosen components to adjacent districts, ensuring districts remain contiguous.
6. **Acceptance Check**: Evaluate the proposed swap using an acceptance probability based on the Metropolis-Hastings criterion.

Metropolis-Hastings Criterion

The **acceptance probability** in the Metropolis-Hastings step ensures that the sampling procedure fairly explores the space of solutions while driving the process toward the desired target distribution. It is defined as:

$$\alpha(\pi', CP \mid \pi) \approx \min\left(1,\ \left(\frac{|B(CP,\pi)|}{|B(CP,\pi')|}\right)^R \frac{F(|B(CP,\pi)|)(1-q)^{|C(\pi',V_{CP})|}}{F(|B(CP,\pi')|)(1-q)^{|C(\pi,V_{CP})|}} \cdot \frac{g(\pi')}{g(\pi)}\right).$$



Variations on Core MCMC

The variations of the MCMC algorithm modify the **Acceptance Check** to handle the **population constraint**:

- **Hard Constraint**: Plans violating the allowed population deviation δ are immediately rejected, strictly enforcing the constraint but limiting mixing efficiency due to frequent rejections.
- **Soft Constraint**: Acceptance is adjusted using a Gibbs distribution to favor near-valid plans. Invalid plans assist transitions and are reweighted later with Sampling-Importance Resampling, improving mixing efficiency.
- **Target Distribution**: Plans can be sampled either uniformly from all contiguous districts or using a Gibbs distribution that emphasizes plans with near-equal populations.