Automated Redistricting Simulation Using GFlowNets

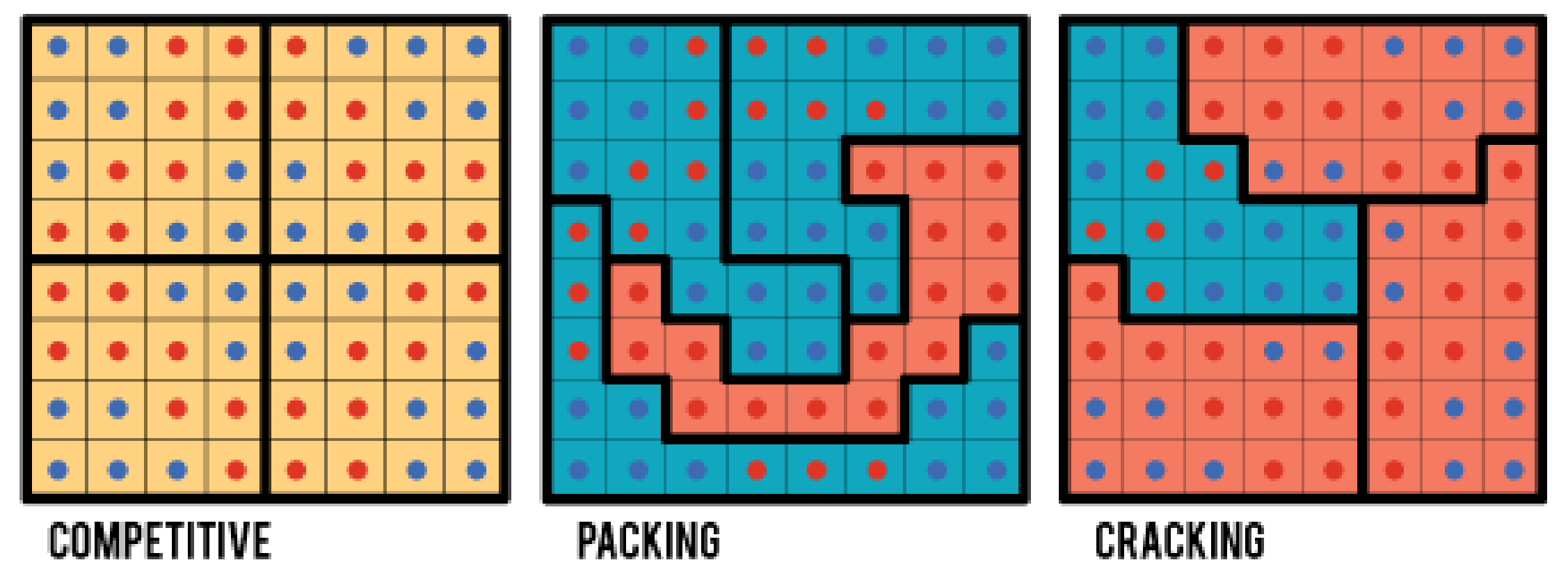
Arnaud Bergeron, Kevin Lessard and Alex Maggioni (alphabetically)

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. :)

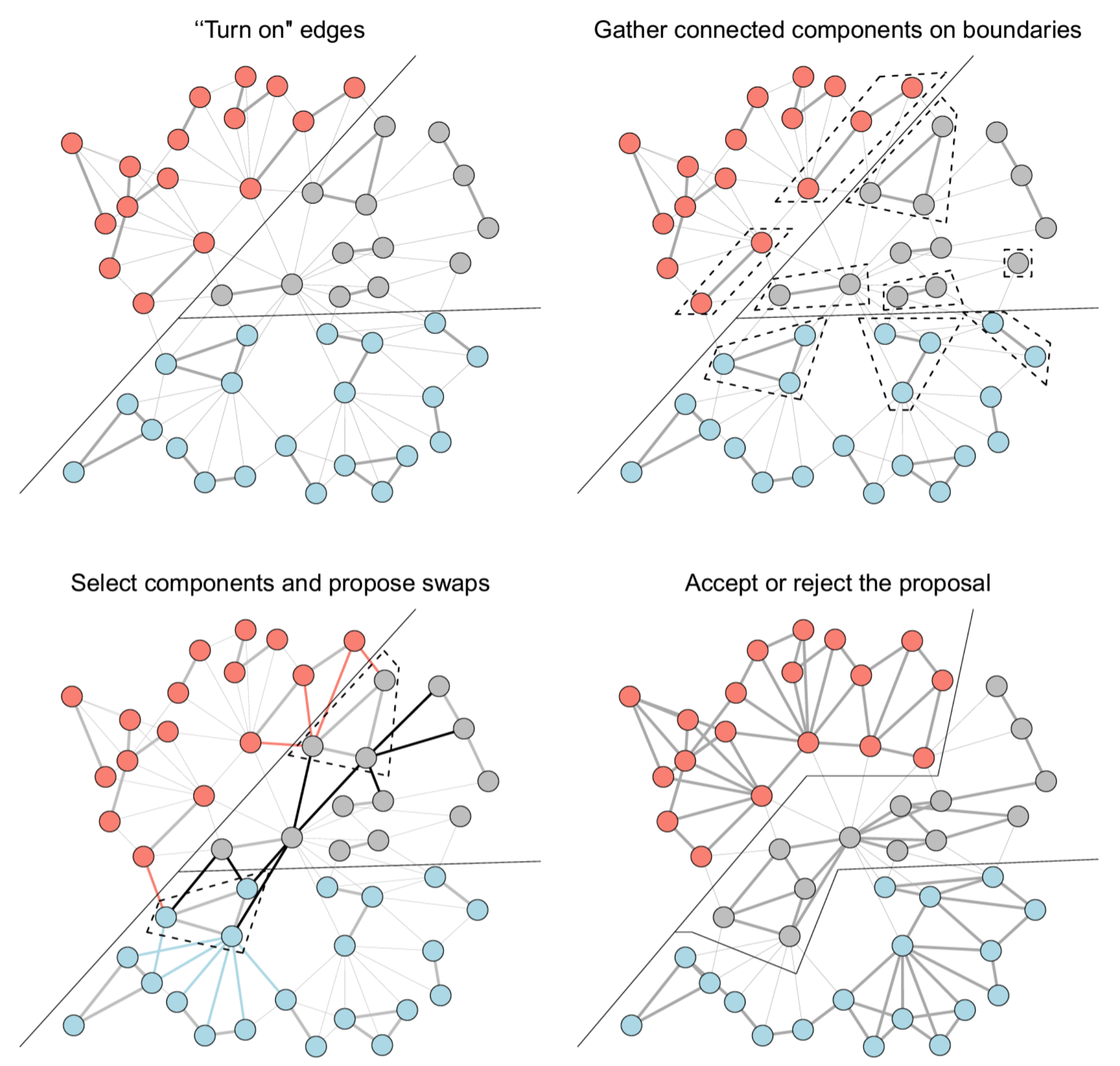
Table 1:Partisan Gap Example

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:

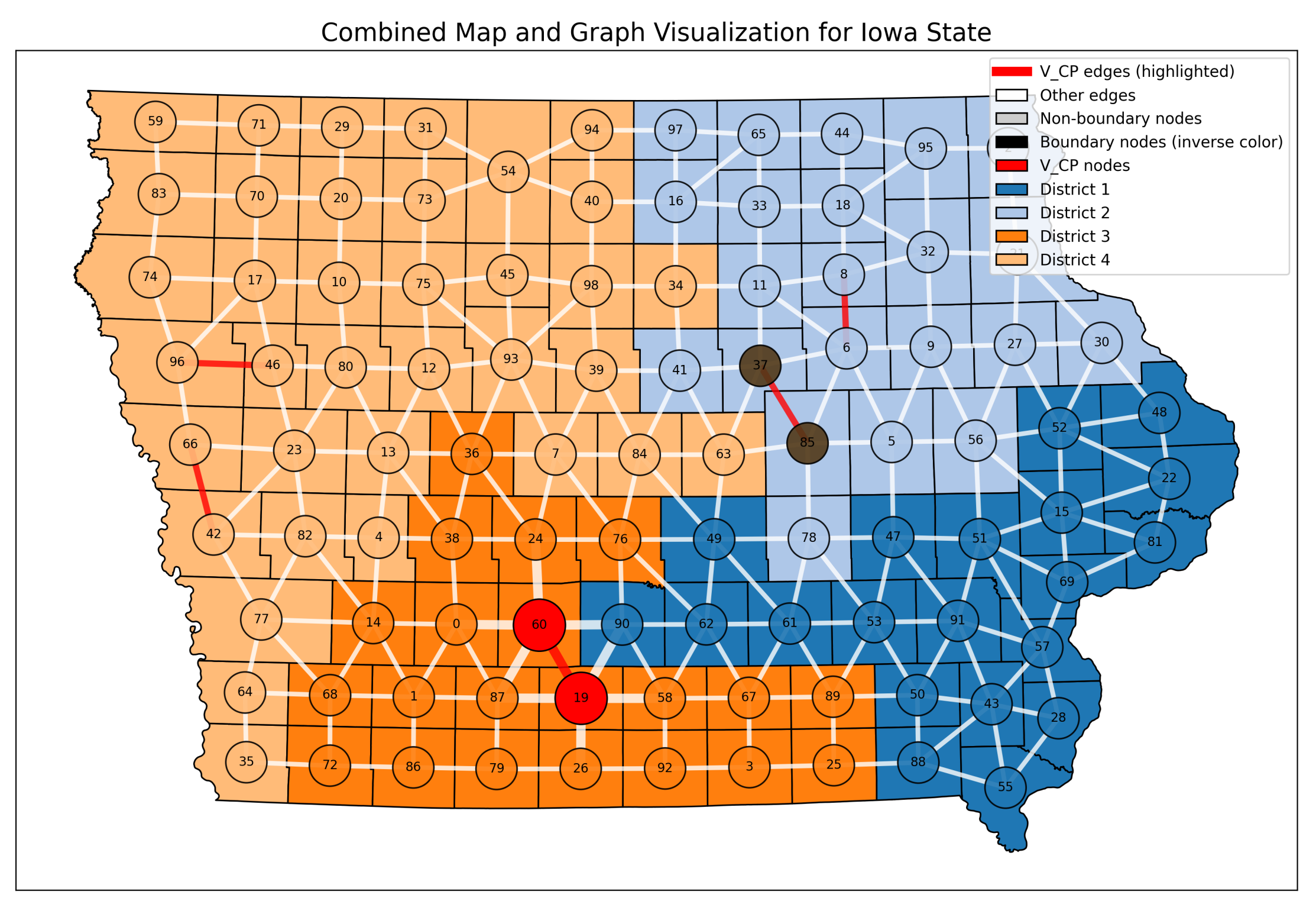

"Turn on" edges
Gather connected components on boundaries
Select components and propose swaps
Accept or reject the proposal

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.

Why GFlowNet?

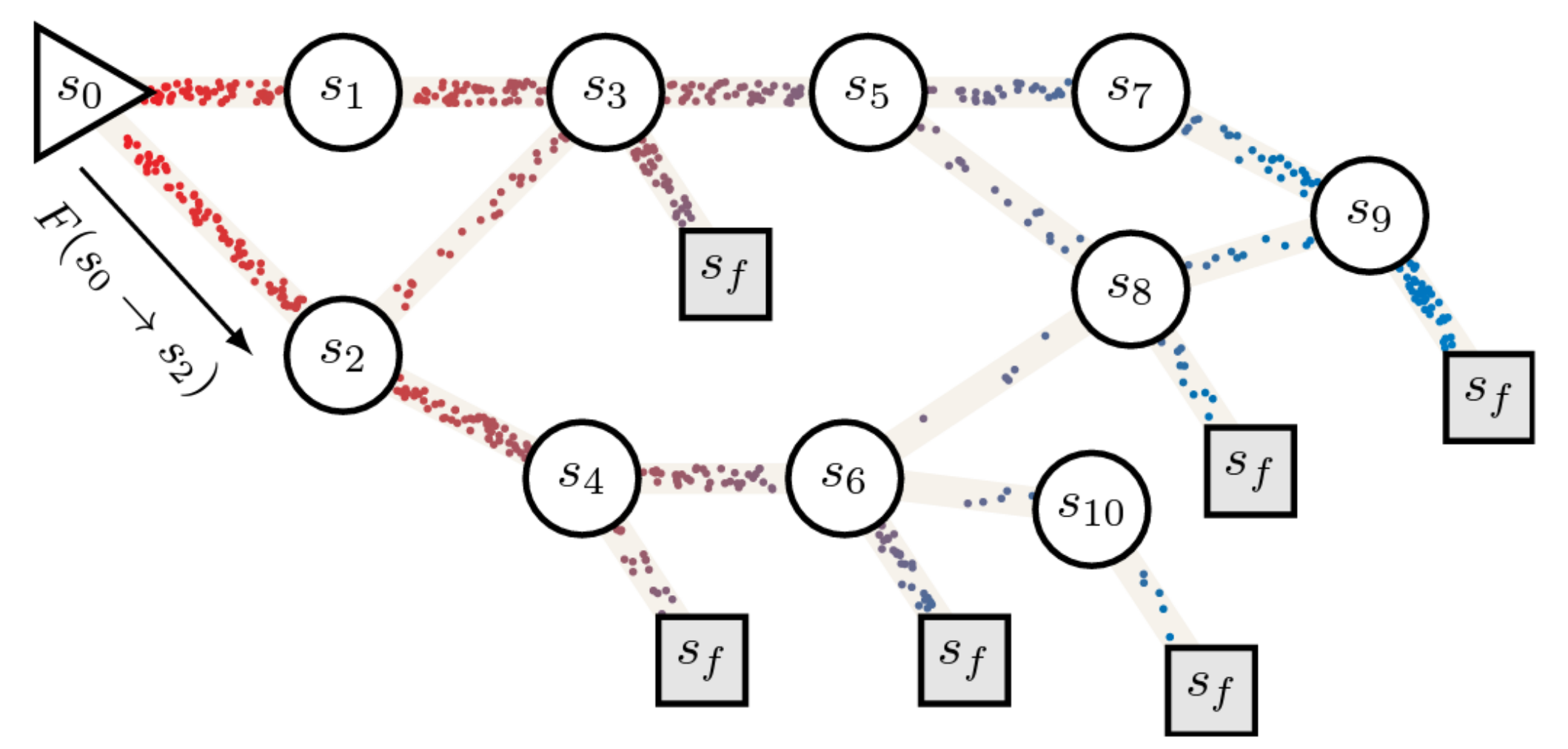
GFlowNets learn a generative policy that produces samples in a single forward pass, avoiding the limitations of long Markov chains in MCMC. Key advantages include:

- **Amortized Sampling:** Once trained, GFlowNets generate independent samples efficiently, without sequential mixing.
- **Diversity:** GFlowNets explore multiple modes of the solution space, producing diverse samples proportional to the reward function $R(s)$.
- **Scalability:** By leveraging neural networks for function approximation, GFlowNets can efficiently scale to large state spaces.

These properties are well suited for our redistricting problem.

GFlowNet Overview

GFlowNets generate structured objects (e.g., redistricting plans) step-by-step by sampling actions through a Directed Acyclic Graph (DAG):



- **States and Actions:** Starting from an initial state s_0, actions transition the system to new states, ending in a final state where the object is complete.
- **Flow Conservation:** The sum of flows into a state equals the sum of flows out, ensuring terminal state probabilities are proportional to their rewards $R(s_f)$.
- **Learning Objective:** GFlowNets optimize a generative policy to match the sampling distribution to the reward function by minimizing a flow-matching loss.
- **Diversity:** Multiple trajectories to terminal states enable exploration of diverse, high-reward solutions.

Design Choices for Our GFlowNet Implementation

State Representation: Each state $\mathcal{S}$ is represented as a tensor of shape:

$$\mathcal{S} \in \mathbb{N}^{N_{\text{counties}} \times (L_{\text{adj}}+1+num\_district)},$$

where N_{counties} is the number of counties, and L_{adj} is the maximum adjacency length.

- Each row corresponds to a county.
- The first L_{adj} entries encode the adjacency information, padded to a fixed length.
- The entry $L_{\text{adj}}+1$ stores the current district assignment.
- The last $num\_district$ entries are to encode valid actions to help the model learn faster.

Counties located at district borders have their district IDs marked negative to highlight valid actions for the MLP.

Action Representation: Actions $\mathcal{A}(s)$ at state s are represented as a vector of shape:

$$\mathcal{A}(s) \in [V_{\text{ID}}, N_{\text{ID}}] \in \mathbb{N}^2.$$

Here, V_{ID} is the county identifier, and N_{ID} is the target district for reassignment.

Ensuring DAG Structure: To enforce acyclic state transitions, we enforce that a county cannot return to a previously assigned district. This ensures that redistricting plans evolve incrementally along a Directed Acyclic Graph (DAG), with each transition forming a valid step toward the final configuration.

Initial State s_0:
The initial state s_0 is the 2020 district partition. Ensuring that all counties are initially assigned to valid districts while preserving adjacency constraints.

Reward Definition

Our reward function is a weighted combination of previously shown evaluation metrics with tunable coefficients λ:

$$\begin{aligned} R_{s_f} &= \lambda_1 \cdot \mathit{Entropy\_pop} \\ &+ \lambda_2 \cdot \mathit{Compactness} \\ &+ \lambda_3 \cdot (1 - |\mathit{Partisan\ Bias}|) \\ &+ \lambda_4 \cdot (1 - |\mathit{Eff\_Gap}|) \end{aligned}$$

Invalid actions are heavily penalized to guide the MLP toward learning valid transitions.

MCMC Results

Partisan Bias vs % Changed Precincts

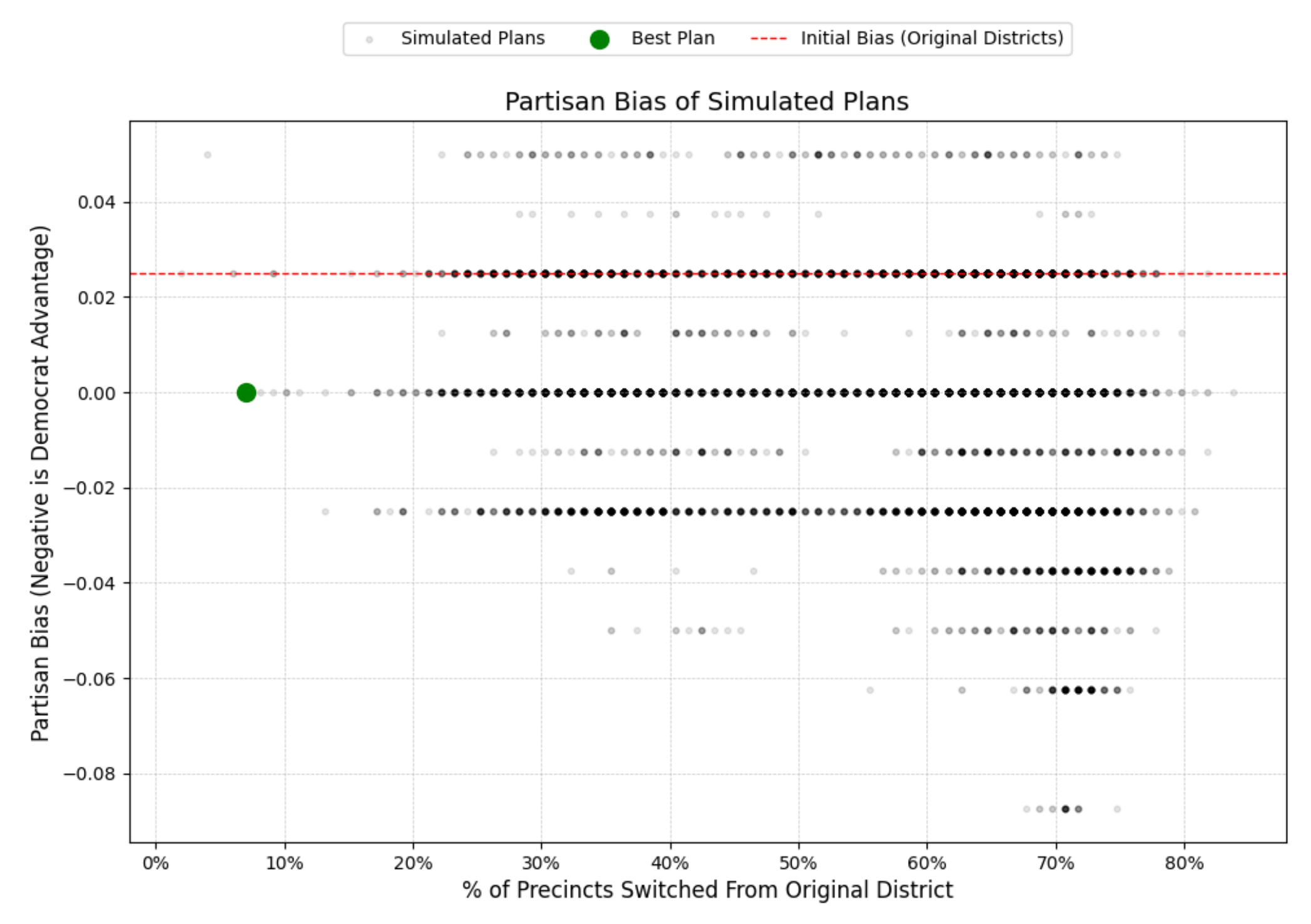


Figure 1: Partisan Bias Across Precinct Swaps

The best plan achieved zero partisan bias (green dot) while the original districts maintained a bias of 0.0250 (red dashed line). while enforcing a minimal change of only 5% of the original presincts.

Map Comparison: MCMC

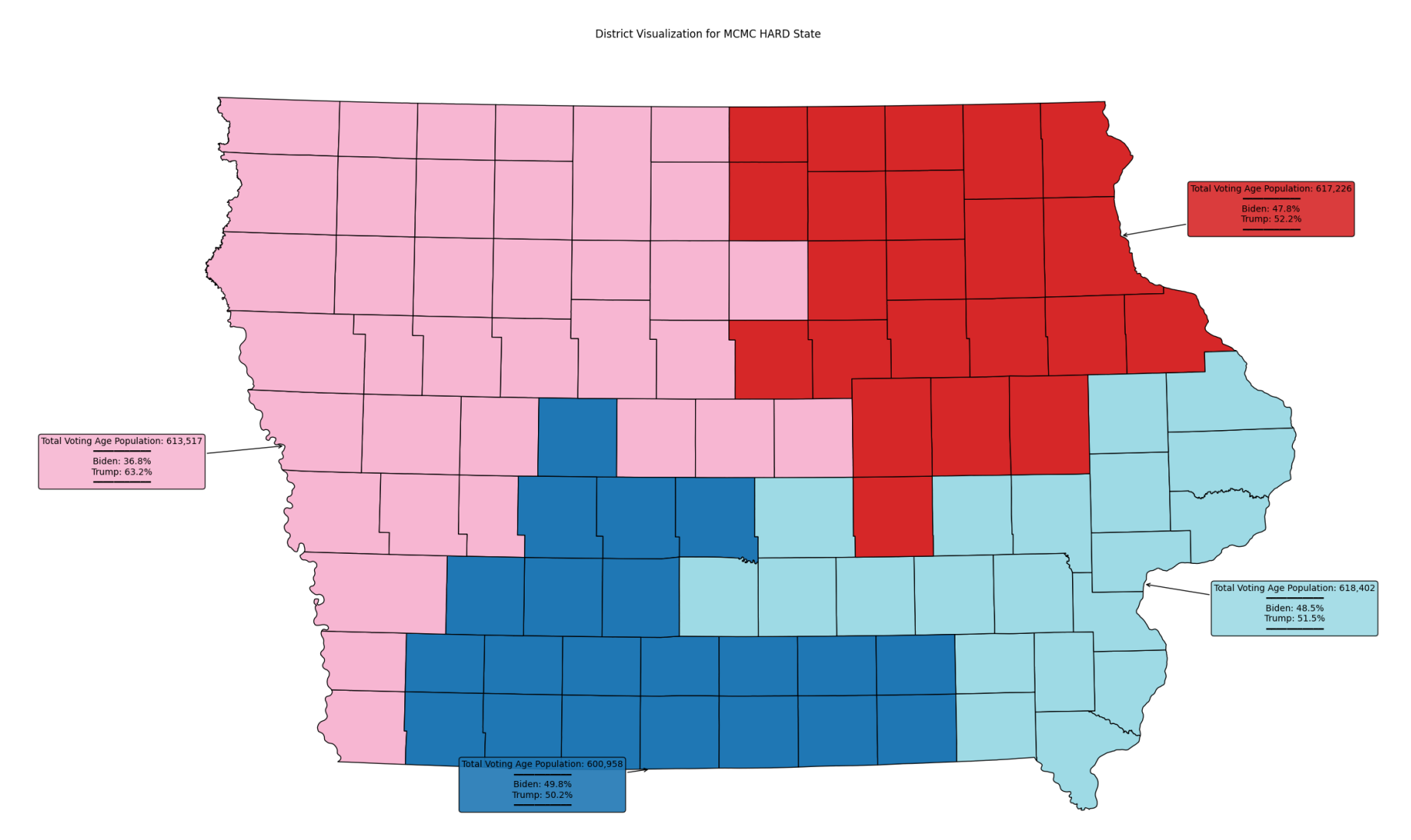


Figure 2: Original District Map (2020)

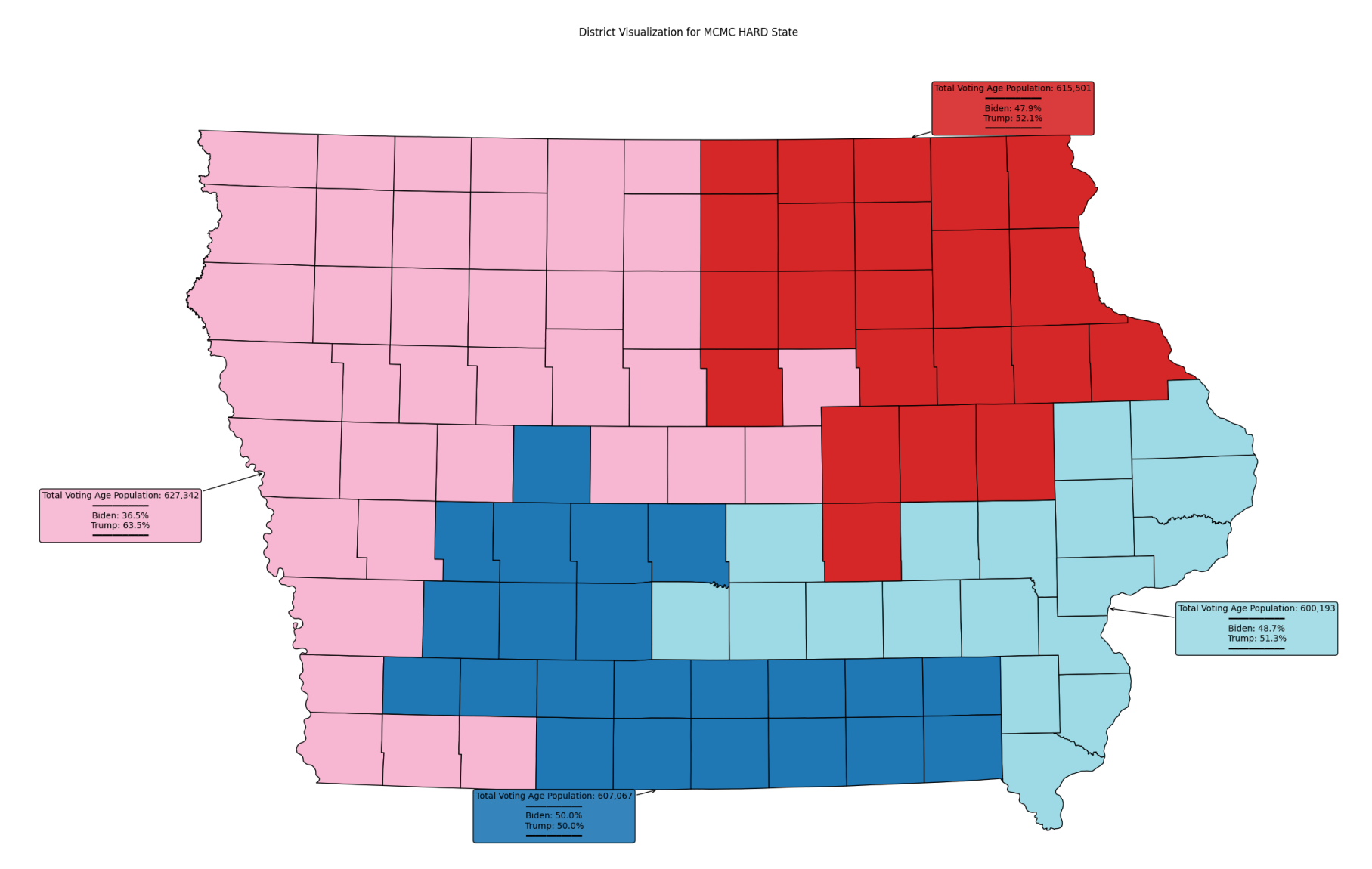


Figure 3: Optimized District Map Using MCMC

Comparison of Metrics:

Metric	Optimized	Original	GFN
Partisan Bias	**0.0000**	0.0250	**0.0000**
Efficiency Gap	0.1641	0.4163	**0.0909**
Mean Compactness	0.2962	0.3334	0.2851
Compactness Std	0.0611	0.0574	0.0555
Max Pop. Deviation	2.8679%	0.0066%	3.535%

Table 1: Comparison of Districting Metrics

GFN Results

We trained our GFN on 128,000 trajectories. The policy samples actions until it picks an invalid one or the exit action. Penalties are given for invalid actions and our reward metrics is computed at each state. We also introduced some probability $\epsilon = 0.03$ of the sampled action to be taken from a uniform distribution over actions.

Map Comparison:GFN

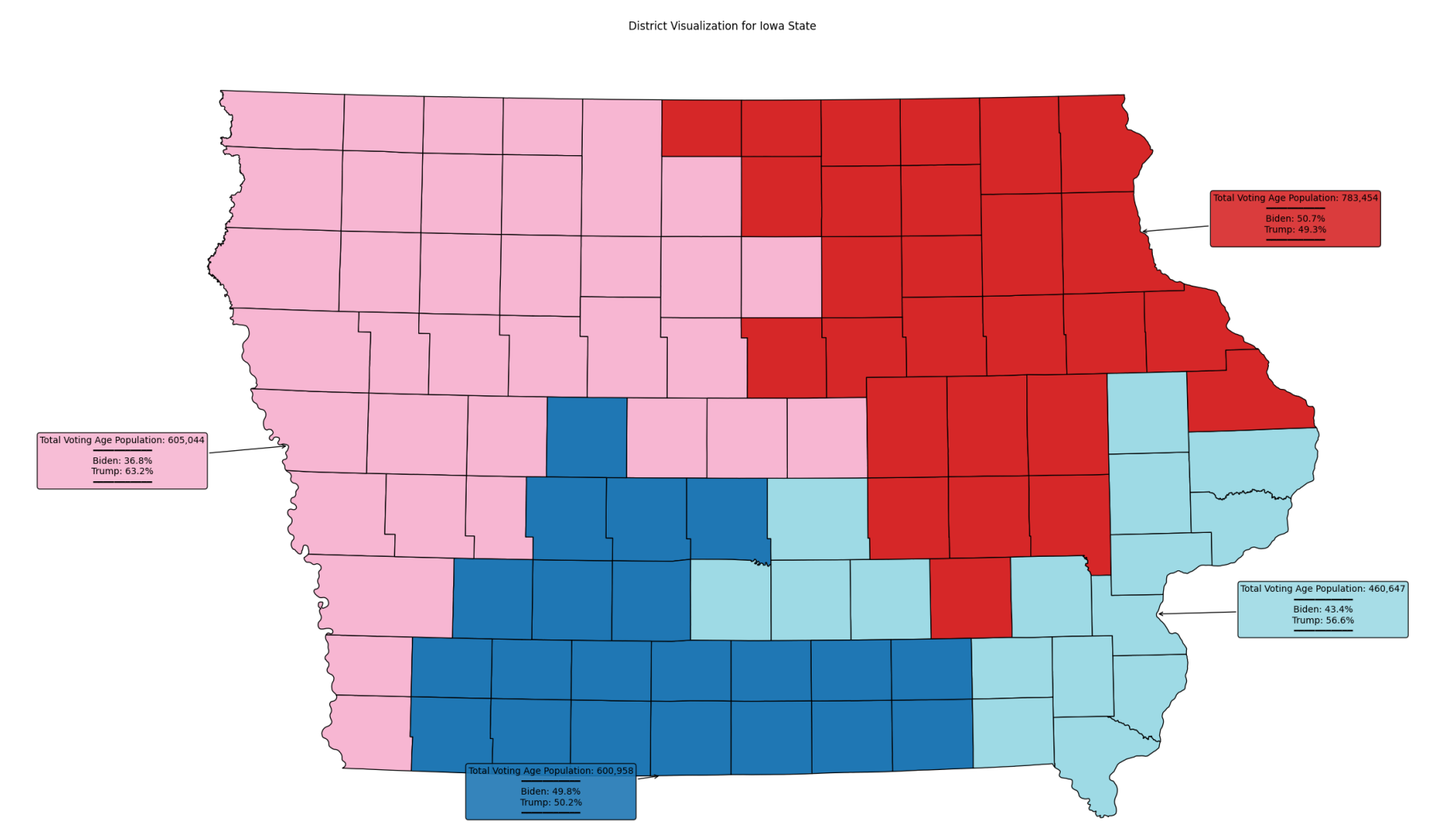


Figure 4: Optimized District Map Using GFN

Our GFlowNet method yields better results than the MCMC and amortizes training so that training the model on an other state would not require to sample as many trajectories to achieve optimal performance.

Possible Improvements

We believe the generalization capability of the GFN could be improved by considering random starting states for each trajectories.

Conclusion

Our results show the advantages of using GFlowNets for graph exploration in the context of Gerrymandering. We demonstrate more efficient exploration that yields to fairer district partitions of electoral counties. These methods could be use to ensure fairness in repartitioning often used to shift political advantage.