The Essential Role of Empirical Validation in Legislative Redistricting Simulation

Kosuke Imai

Harvard University

Quantitative Redistricting and Gerrymandering Conference
Duke University
March 3, 2020

Joint work with Ben Fifield, Jun Kawahara (Kyoto) and Chris Kenny

Motivation

- Congressional redistricting as a key element of American democracy
- Influenced by political motives and partisan ends
- Early proposals in 1960s: automated simulation as a transparent, objective, and unbiased method for redistricting

- Resurgence of simulation methods over the last 20 years
  - increasing availability of granular data about voters
  - recent advances in computing capability and methods
- Starting to be used in courts (e.g., MO, NC, and OH)

- Do simulation methods can actually yield a representative sample of all possible redistricting plans that satisfy required constraints?

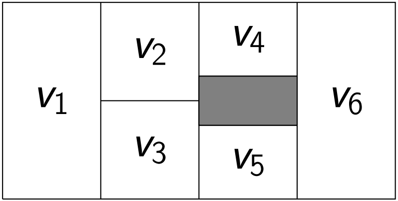
Overview

- Insufficient efforts have been made to empirically validate redistricting simulation methods
- Eric Lander in his amicus brief to the Supreme Court:

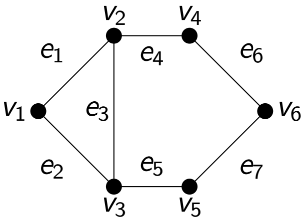
  With modern computer technology, it is now straightforward to generate a large collection of redistricting plans that are representative of all possible plans that meet the State's declared goals (e.g., compactness and contiguity)

- Some used 25 precinct validation set of Fifield *et al.* (forthcoming)
- We apply the computational method of Kawahara *et al.* (2017)
  1. efficiently enumerate all possible redistricting plans
  2. independently and uniformly sample from this population
- Scales to a state with a couple of hundred geographical units
  1. enumeration: a large number of small validation sets
  2. sampling: a small number of medium-size validation sets

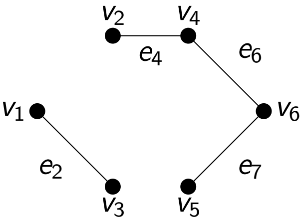
Redistricting as a Graph-partitioning Problem



(a) Original map



(b) Graph representation

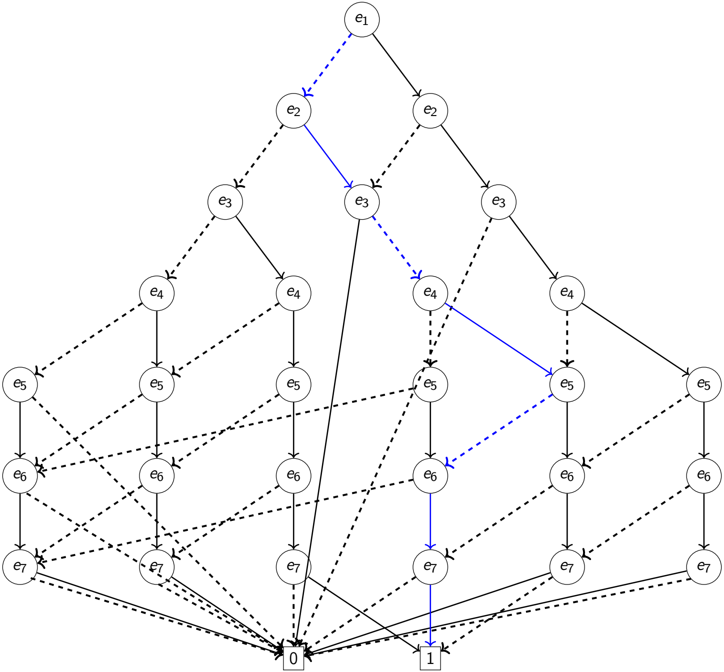


(c) Induced subgraphs

Zero-suppressed Binary Decision Diagram (ZDD)

- A data structure to efficiently represent a family of sets (Minato, 1993)
- ZDD as a directed acyclic graph
  - *root node* e_1: no incoming arc, represents an edge of the original graph
  - *terminal nodes*: no outgoing arc, no correspondence to an edge of the original graph
    1. *0-terminal* $\boxed{0}$
    2. *1-terminal* $\boxed{1}$
  - every non-terminal node has two outgoing arcs
    1. *0-arc*: dashed arc $\dashrightarrow$ removes edge
    2. *1-arc*: solid arc $\longrightarrow$ retains edge
- One-to-one correspondence:
  - Graph partition: $\{e_2, e_4, e_6, e_7\}$
  - The set of edges that belong to a directed path from the root node to 1-terminal node and have an outgoing 1-arc

$$e_1 \dashrightarrow e_2 \longrightarrow e_3 \dashrightarrow e_4 \longrightarrow e_5 \dashrightarrow e_6 \longrightarrow e_7 \longrightarrow \boxed{1}$$


e_1
e_2
e_2
e_3
e_3
e_3
e_4
e_4
e_4
e_4
e_5
e_5
e_5
e_5
e_5
e_5
e_6
e_6
e_6
e_6
e_6
e_6
e_7
e_7
e_7
e_7
e_7
e_7
0
1

Construction of ZDD

- Starting with root node e_1, create one outgoing 0-arc and one outgoing 1-arc from one node e_ℓ to the next node $e_{\ell+1}$
- Store the number of determined connected components or $\texttt{dcc}$ for each node: $\texttt{dcc} = 1$ for e_5

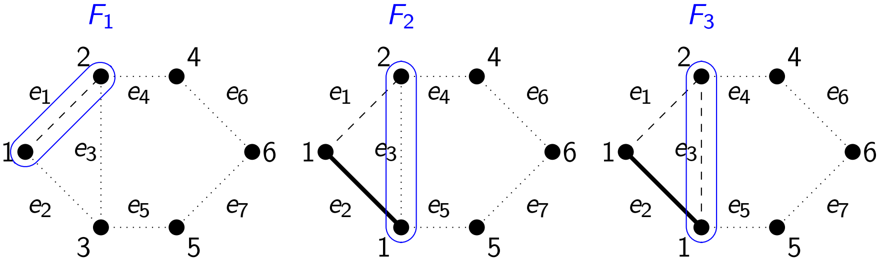
$$e_1 \longrightarrow e_2 \dashrightarrow e_3 \dashrightarrow e_4 \dashrightarrow e_5$$

$\rightsquigarrow \{v_1, v_2\}$ forms a district regardless of whether or not e_5 is retained

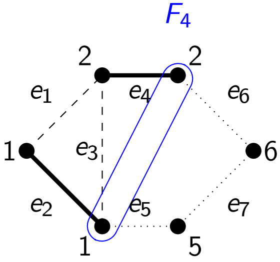
- Create an arc into the 0-terminal node if $\texttt{dcc} > p$ at any node or $\texttt{dcc} < p$ at the final node
- Keep track of connected component number for each vertex of the original graph
  1. Start by setting $\texttt{comp}[v_i] \leftarrow i$ for $i = 1, 2, \ldots, n$
  2. If we retain an edge between v_i and $v_{i'}$, then set $\texttt{comp}[v_j] \leftarrow \min\{\texttt{comp}[v_i], \texttt{comp}[v_{i'}]\}$ for any v_j with $\texttt{comp}[v_j] = \max\{\texttt{comp}[v_i], \texttt{comp}[v_{i'}]\}$

The Frontier-based Search

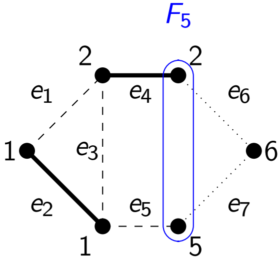
- Frontier: the set of vertices of the original graph that are incident to both a processed edge and an unprocessed edge
- $F_0 = F_m = \emptyset$ where m is the total number of edges
- Frontier can be used to determine a connected component number
  - suppose there exists a vertex v such that $v \in F_{\ell-1}$ but $v \notin F_\ell$
  - If there is no other vertex in F_ℓ shares the connected component number, then comp[v] is determined and dcc is incremented by 1



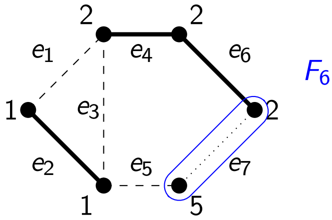
(a) Process edge e_1; dcc $= 0$ (b) Process edge e_2; dcc $= 0$ (c) Process edge e_3; dcc $= 0$



(d) Process edge e_4; $\texttt{dcc} = 0$



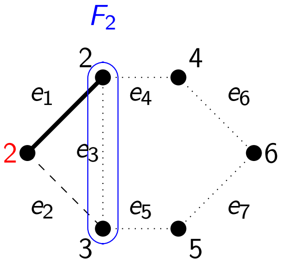
(e) Process edge e_5; $\texttt{dcc} = 1$



(f) Process edge e_6; $\texttt{dcc} = 1$

Node Merge for Computational Efficiency

- Only required information = connectivity of vertices in $F_{\ell-1}$
- Merge multiple paths at node e_ℓ if `dcc` and the frontier $F_{\ell-1}$ are identical after renumbering to eliminate gaps



(a) Path $e_1 \longrightarrow e_2 \dashrightarrow e_3$ (b) Path $e_1 \dashrightarrow e_2 \longrightarrow e_3$

- Merging is critical: 8×8 lattice into 2 districts $\approx 1.2 \times 10^{11}$ partitions
- Can encode the population parity and other information into ZDD $\rightsquigarrow$ prevents merging and hence does not scale

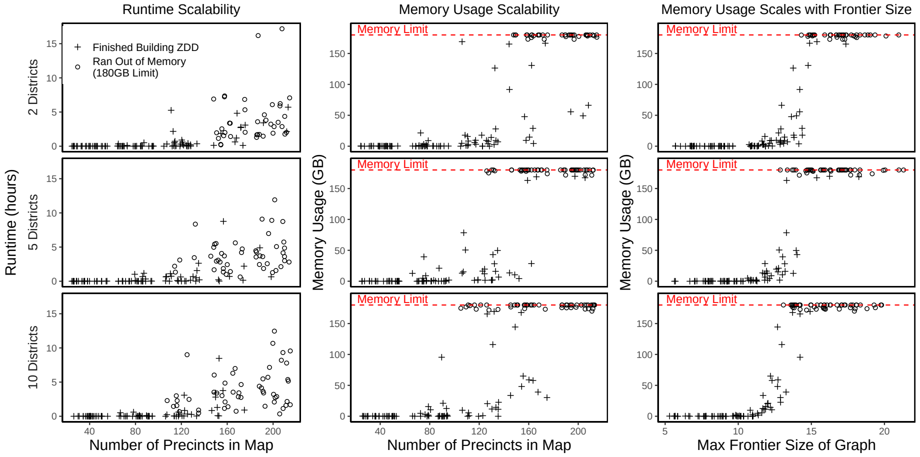
Enumeration and Independent Sampling

- Every path from the root node to the 1-terminal node has one-to-one correspondence to a graph partition
- Enumerate all the paths
  1. start with the 1-terminal node
  2. count the number of unique paths at each node
  3. move upwards until the root node is reached

- Independent sampling (Knuth 2011)
  - Let $c(e_\ell)$ be the number of paths from the 1-terminal node to node e_ℓ
  - Let e_{ℓ_0} and e_{ℓ_1} be the nodes pointed by the 0-arc and 1-arc of e_ℓ
  - Store $c(e_\ell)$ for each node e_ℓ
  - Conduct random sampling by starting with the root node and choosing node e_{ℓ_1} with probability $c(e_{\ell_1})/\{c(e_{\ell_1}) + c(e_{\ell_0})\}$
  - Probability of reaching the 0-terminal node is zero

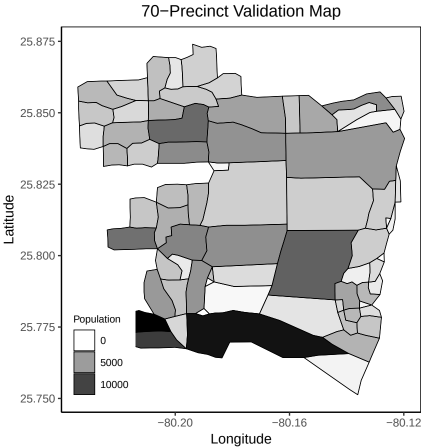
Scalability of the ZDD Construction Algorithm

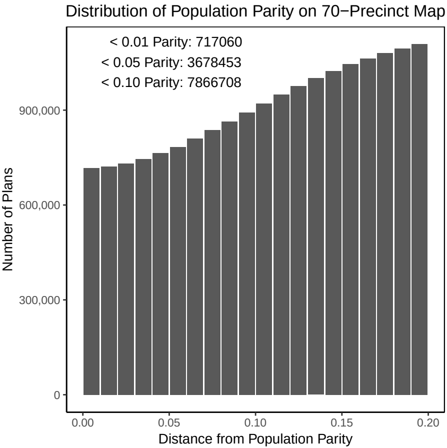
- Randomly generate contiguous subsets of the New Hampshire map (327 precincts and 2 districts) that vary in size $\{40, 80, \ldots, 200\}$
- Number of districts: 2, 5, or 10
- Cluster with 530 nodes, 48 cores and 180 GB of RAM per node



Validation through Enumeration

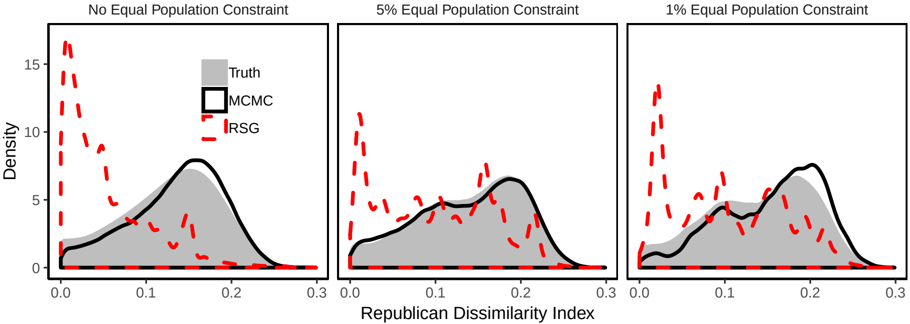
- 70 precinct validation set from Florida ($\gg$ 25 validation set)
- 8 hours on MacBook Pro laptop with 16GB and 2.8 GHz processor
- Building ZDD took less than a second: 44 million valid plans





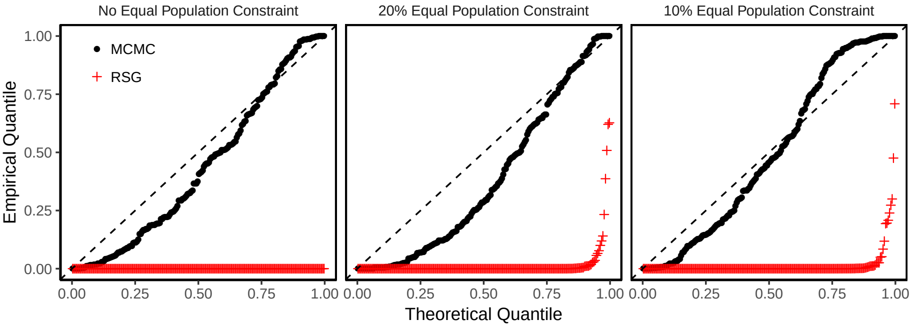
Performance of RSG and MCMC Algorithms

- Evaluate the performance of two common algorithms:
  1. Random-seed-and-grow (RSG): Cirincione *et al.* (2000); Chen and Rodden (2013)
  2. Markov chain Monte Carlo (MCMC) Fifield *et al.* (2014); Mattingly and Vaughn (2014)
- Implemented via the `redist` package
- tempering/discarding/reweighting for population constraints
- Republican dissimilarity index as a test statistic



Many Small Validation Maps

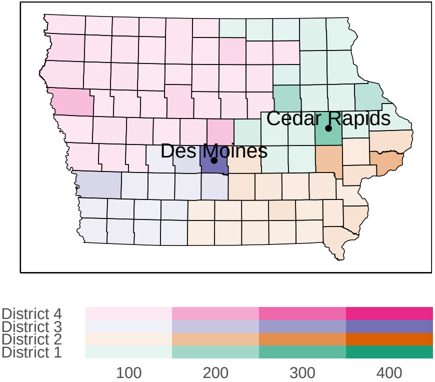
- Robustness to many different maps
- No separate tuning or convergence diagnostics for each map
- Setup
  1. 200 independent 25-precinct sets from Florida
  2. 5 million iterations, taking every 500th draw
  3. conduct the Kolmogorov-Smirnov test and record the *p*-value



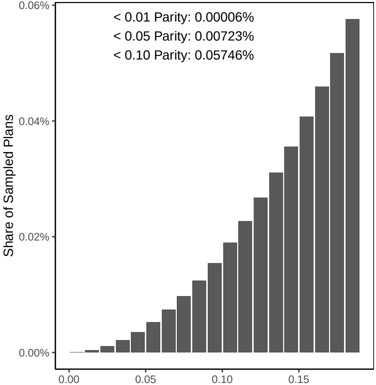
Validation through Independent Uniform Sampling

- Even if we can build ZDD, enumeration is computationally intensive
- Random sampling addresses this issue via Monte Carlo approximation
- Iowa map: 4 districts
  - 99 counties; no county is supposed to be split
  - 500 million independent draws
  - actual map has a population parity of less than 0.0001

Congressional Districts of Iowa (2016)

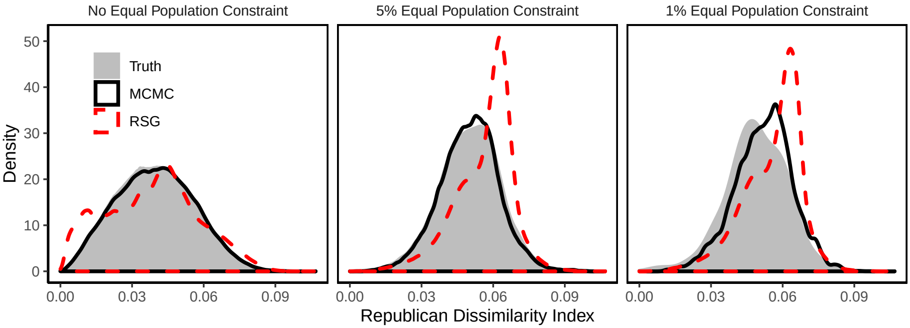


Distribution of Population Parity on Iowa Map

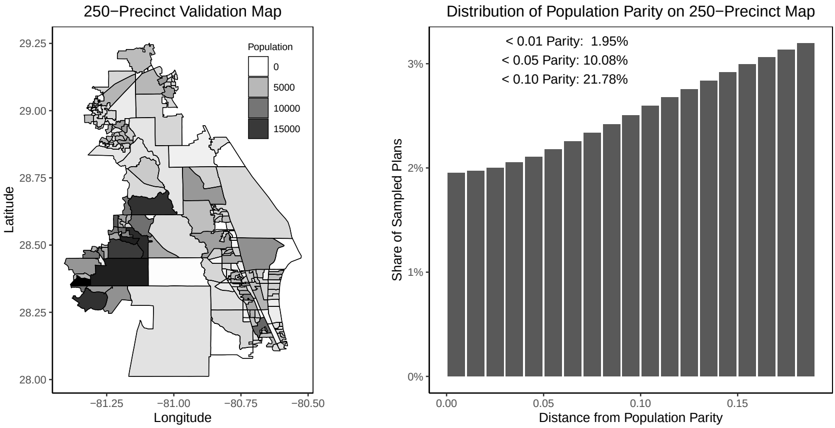


Performance for the Iowa Validation Map

- MCMC:
  - 8 chains with 250K iterations each
  - Gelman-Rubin diagnostics indicates convergence after 30K iterations
  - 5% parity: 630K maps
  - 1% parity: 93K maps
- RSG: 2 million independent draws



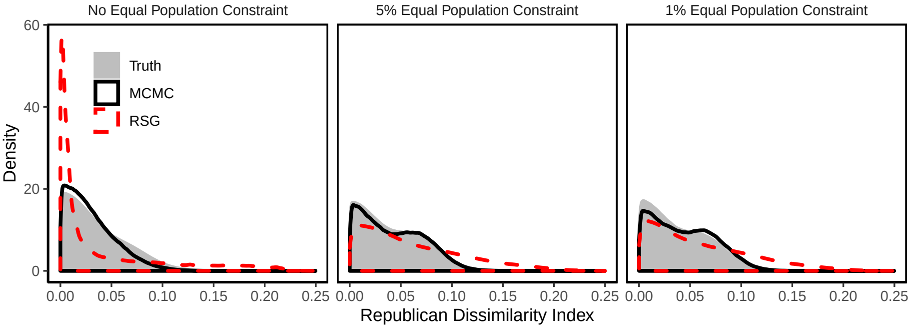
A New 250-Precinct Validation Map



- The largest validation map taken from Florida
- 2 districts $\rightsquigarrow$ total number of contiguous plans $= 5^{39}$
- We uniformly sample 100 million partitions

Empirical Performance

- MCMC:
  - 8 chains with 500K iterations each
  - Gelman-Rubin diagnostics indicates convergence after 75K iterations
  - 5% parity: 3 million maps
  - 1% parity: 1.9 million maps
- RSG: 4 million independent draws



Concluding Remarks

- Increasing use of computational methods to generate redistricting plans in legislatures and determine their legality in courts
- Scientific community must empirically validate the performance of various proposed simulation methods

- We apply the recently developed enumeration method
  1. more realistic validation maps including Iowa and a 250-precinct map
  2. an MCMC algorithm significantly outperforms a RSG algorithm
  3. the algorithm and validation maps will be made available

- Ongoing work:
  1. consequences of various constraints other than contiguity and population parity
  2. further scaling up the enumeration algorithm

References

- Fifield, Imai, Kawahara, and Kenny. (2020). "The Essential Role of Empirical Validation in Legislative Redistricting Simulation." *Working Paper*
- Fifield, Higgins, Imai, and Tarr. "Automated Redistricting Simulation Using Markov Chain Monte Carlo." *Journal of Computational and Graphical Statistics*, Forthcoming.
- `redist`: Markov Chain Monte Carlo Methods for Redistricting Simulation. available at CRAN

https://imai.fas.harvard.edu