Technical Document: Agent-Based Polarization Simulation Framework

Emre Erdoğan, Pınar Uyan-Semerci, Ayça Ebru Giritligil,
Onur Doğan, Giray Girengir
April 2025

Introduction

This document provides a technical overview of the agent-based polarization simulation implemented in the Streamlit application *app_final_v06.py*. The simulation models opinion dynamics in a 2D opinion space, where agents interact based on preference profiles, evolve their opinions over time, and influence polarization and voting outcomes. The framework is designed to study social choice dynamics and polarization from a computational perspective, drawing on theories of social contract and democratic processes.

The simulation supports:

- **Agent Interactions**: Agents adjust their positions and preferences based on interactions governed by a delta matrix.
- **Polarization Metrics**: Four metrics (Party, Preference, Binary, Kemeny) are calculated to quantify polarization.
- **Social Choice Outcomes**: Voting results and social choice winners (Plurality, Borda, Majority Comparison, Copeland) are computed.
- **Interactive Visualizations**: Results are displayed using Plotly animations and charts within a Streamlit interface.

Model Framework

Agents and Opinion Space

- **Agents**: Represented by the *Agents* class (defined in *agents.py*), each agent has a position in a 2D opinion space and a preference profile.
- **Opinion Space**: A square 2D space defined by *OPINION_SPACE_SIZE* (default: 10), where positions are coordinates $(x, y) \in [-\frac{\text{OPINION\_SPACE\_SIZE}}{2}, \frac{\text{OPINION\_SPACE\_SIZE}}{2}]^2$.
- **Preference Profiles**: For N_{PARTIES} parties, there are $N_{\text{PROFILES}} = N_{\text{PARTIES}}!$ possible preference profiles, indexed from 0 to $N_{\text{PROFILES}} - 1$.

Interaction Rules

Agents interact pairwise based on a delta matrix (Δ), which defines the probability of interaction between agents with different preference profiles. The interaction process involves:

1. **Attraction**: Agents move closer to each other based on a similarity-driven attraction force.
2. **Reaction**: Agents adjust their positions relative to party positions based on their preferences.

Mathematical Formulations

Agent Position Update

The position update for agent i at iteration t is governed by the deliberation step (implemented in *deliberation.py*). Let:

- $\mathbf{p}_i(t) = (x_i(t), y_i(t))$ be the position of agent i at iteration t.
- $\text{pref}_i(t)$ be the preference profile index of agent i.
- $\Delta_{\text{pref}_i,\text{pref}_j}$ be the interaction probability between agents with profiles pref_i and pref_j.
- $\mathbf{P}_k = (P_{k,x}, P_{k,y})$ be the position of party k.

The update rule for agent i paired with agent j is:

$$\mathbf{p}_i(t+1) = \mathbf{p}_i(t) + \mu_a \cdot (\mathbf{p}_j(t) - \mathbf{p}_i(t)) \cdot \text{discount}(t) + \mu_r \cdot (\mathbf{P}_{k_i} - \mathbf{p}_i(t))$$

Where:

- $\mu_a =$ MU_ATTRACTION (default: 0.1) is the attraction coefficient.
- $\mu_r =$ MU_REACTION (default: 0.05) is the reaction coefficient.
- $\text{discount}(t) =$ DISCOUNT_COEFF $\cdot (1 - \frac{t}{T})$ (default: DISCOUNT_COEFF $= 1.0$) reduces attraction over time.
- k_i is the first-choice party of agent i.
- T is the total number of iterations.

Preference Update

Agent preferences are updated dynamically based on their proximity to party positions. For agent i, the preference ranking is determined by sorting the Euclidean distances to each party:

$$d_{i,k} = \sqrt{(x_i - P_{k,x})^2 + (y_i - P_{k,y})^2}$$

The preference profile is the permutation of party indices $(k_1, k_2, \ldots, k_{N_{\text{PARTIES}}})$ such that $d_{i,k_1} \leq d_{i,k_2} \leq \cdots \leq d_{i,k_{N_{\text{PARTIES}}}}$. The first-choice party is k_1.

Polarization Metrics

Four polarization metrics are calculated (implemented in *polarisation.py*):

1. **Party Polarisation**: Measures the average distance between party centers (mean positions of agents supporting each party):

$$\text{PartyPolarisation} = \frac{1}{N_{\text{PARTIES}}(N_{\text{PARTIES}} - 1)} \sum_{k \neq l} \sqrt{(C_{k,x} - C_{l,x})^2 + (C_{k,y} - C_{l,y})^2}$$

   Where $C_k = (C_{k,x}, C_{k,y})$ is the center of party k, computed as the mean position of agents with first-choice party k.

2. **Preference Polarisation**: Measures the variance of preference profile indices across agents:

$$\text{PrefPolarisation} = \frac{1}{N_{\text{AGENTS}}} \sum_{i=1}^{N_{\text{AGENTS}}} (\text{pref}_i - \bar{\text{pref}})^2$$

   Where $\bar{\text{pref}}$ is the mean preference profile index.

3. **Binary Polarisation**: Assumes a binary split of agents (e.g., based on the two largest parties) and measures the distance between their centers:

$$\text{BinaryPolarisation} = \sqrt{(C_{m,x} - C_{n,x})^2 + (C_{m,y} - C_{n,y})^2}$$

   Where C_m and C_n are the centers of the two largest parties by support.

4. **Kemeny Polarisation**: Measures disagreement in preference rankings using the Kemeny distance between preference profiles. For agents i and j:

$$\text{KemenyDistance}(\text{pref}_i, \text{pref}_j) = \sum_{k,l} \mathbb{I}\,(\text{order}_{i,k,l} \neq \text{order}_{j,k,l})$$

   Where $\text{order}_{i,k,l} = 1$ if agent i prefers party k over l, and 0 otherwise. The total Kemeny Polarisation is:

$$\text{KemenyPolarisation} = \frac{1}{N_{\text{AGENTS}}(N_{\text{AGENTS}} - 1)} \sum_{i \neq j} \text{KemenyDistance}\,(\text{pref}_i, \text{pref}_j)$$

Social Choice Outcomes

Social choice winners are computed for four rules (implemented in *`polarisation.py`*):

1. **Plurality**: The party with the most first-choice votes.

$$\text{PluralityVotes}_k = \sum_{i=1}^{N_{\text{AGENTS}}} \mathbb{I}\,(\text{first\_choice}_i = k)$$

$$\text{PluralityWinner} = \underset{k}{\text{argmax}}\,\text{PluralityVotes}_k$$

2. **Borda**: Each agent assigns points to parties based on their ranking (e.g., for 3 parties, 2 points for first, 1 for second, 0 for third).

$$\text{BordaScore}_k = \sum_{i=1}^{N_{\text{AGENTS}}} (N_{\text{PARTIES}} - \text{rank}_{i,k})$$

$$\text{BordaWinner} = \underset{k}{\text{argmax}} \text{BordaScore}_k$$

3. **Majority Comparison**: Compares pairs of parties, awarding points based on pairwise majority preferences.

$$\text{MajCompScore}_k = \sum_{l \neq k} \mathbb{I}\,(\text{num}_{k>l} > \text{num}_{l>k})$$

Where $\text{num}_{k>l}$ is the number of agents preferring k over l.

$$\text{MajCompWinner} = \underset{k}{\text{argmax}} \text{MajCompScore}_k$$

4. **Copeland**: Awards points based on pairwise wins, with a tie giving 0.5 points.

$$\text{CopelandScore}_k = \sum_{l \neq k} \begin{cases} 1 & \text{if } \text{num}_{k>l} > \text{num}_{l>k} \\ 0.5 & \text{if } \text{num}_{k>l} = \text{num}_{l>k} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{CopelandWinner} = \underset{k}{\text{argmax}} \text{CopelandScore}_k$$

Simulation Mechanics

Simulation Loop

The simulation runs for T iterations, recording data at intervals to create N_{FRAMES} animation frames:

- **Step Interval**: $\text{step\_interval} = \max(1, \lfloor T / N_{\text{FRAMES}} \rfloor)$.
- **Record Iterations**: Data is recorded at iterations $t = \text{step\_interval}, 2 \cdot \text{step\_interval}, \ldots, T$.

For each iteration t:

1. Select pairs of agents to interact based on Δ.
2. Update agent positions using the position update rule.
3. Update agent preferences based on distances to party positions.
4. If t is a record iteration, compute:
    - Agent positions, first-choice parties, and preference profiles.
    - Party centers and society center.
    - Polarization metrics.

- Voting shares and social choice outcomes.

Visualization Logic

- **Scatter Animation**: Uses Plotly to create an interactive animation with:
  - Agents as colored dots (color by first-choice party).
  - Party positions as stars, party centers as diamonds, society center as a black square.
  - Frames correspond to recorded iterations, with play/pause controls.
- **Polarization Metrics**: Line plots showing the four metrics over time, with a dropdown to switch between metrics.
- **Voting Results**: Includes line plots for party support, bar charts for final support, and tables for social choice winners.

Data Structures

Key Variables

- **Agent Positions**: `agents.positions`, a NumPy array of shape $(N_{\text{AGENTS}}, 2)$ storing (x, y) coordinates.
- **Preference Profiles**: `agents.pref_indices`, a NumPy array of shape $(N_{\text{AGENTS}},)$ storing profile indices.
- **Delta Matrix**: `delta_matrix`, a NumPy array of shape $(N_{\text{PROFILES}}, N_{\text{PROFILES}})$, defining interaction probabilities.
- **Party Positions**: `party_positions`, a NumPy array of shape $(N_{\text{PARTIES}}, 2)$, storing (x, y) coordinates of parties.
- **Simulation Records**:
  - `positions_record`: List of tuples $(frame, frame_{p}arty_{c}enters, frame_{s}ociety_{c}enter)$, where each frame is a Pandas DataFrame.
  - `polarisation_df`: DataFrame with columns `["Iteration", "PartyPolarisation", "PrefPolarisation", "BinaryPolarisation", "KemenyPolarisation"]`.
  - `voting_df`: DataFrame with columns `["Iteration", 0, 1, ..., N_PARTIES-1]`, storing voting shares.
  - `social_choice_df`: DataFrame with columns for winners and scores under each rule.

Methodology

Simulation Setup

- **Parameters**:
    - N_{AGENTS}: Number of agents (50–2000, default: 200).
    - T: Number of iterations (50–1000, default: 300).
    - N_{FRAMES}: Number of animation frames (10–100, default: 30).
    - N_{PARTIES}: Number of parties (2–5, default: 3).
- **Scenarios**: Predefined scenarios (in *`scenarios.py`*) provide *`delta_matrix`* and *`party_positions`*. Users can also manually define party positions.

Data Collection

- Agent positions and preferences are recorded at each frame.
- Polarization metrics and voting outcomes are computed at each recorded iteration.
- Results are stored in memory and can be saved/exported as Excel files.

Visualization and Analysis

- Plotly is used for interactive visualizations, with Streamlit providing the user interface.
- Users can save multiple simulation runs and export selected results, with metadata and per-simulation sheets.

Assumptions and Limitations

Assumptions

- Agents interact homogeneously within the 2D opinion space, with no external influences (e.g., media, events).
- The delta matrix is static, assuming constant interaction probabilities.
- Preference updates are based solely on Euclidean distances to party positions, ignoring other factors like ideology or social networks.
- Polarization metrics assume linear relationships in some cases (e.g., Party Polarisation), which may oversimplify complex dynamics.

Limitations

- **Scalability**: Large values of N_{AGENTS} or T can lead to slow performance due to computational complexity.
- **Simplification**: The 2D opinion space may not fully capture multidimensional opinion structures.

- **Static Parties**: Party positions are fixed, which doesn't reflect real-world dynamics where parties may shift positions.
- **Binary Polarisation**: Assumes a two-party dominance, which may not apply to multiparty systems with fragmented support.

developed by Emre Erdoğan, Pınar Uyan-Semerci, Ayça Ebru Giritligil, Onur Doğan & Giray Girengir—2025