Hopfield and Hebbian Models of Belief Polarization: Neural Networks, Social Contexts

Graham Sack, Carissa Flocken, Patrick Grim, Aaron Bramson, William Berger, Daniel J. Singer, and Steven Fisher

Abstract. In this paper, the authors develop two models of belief polarization on signed social networks. First, we consider a Hopfield network model for opinion polarization. Although Hopfield networks were originally proposed in a neural context as a means of generating content addressable memory, we propose that the mechanism and algorithms also have a plausible interpretation in the context of social networks of belief and trust, leading to several insights about stable states and basins of attraction in polarized communities. The Hopfield model assumes one-way causality, whereby an agent's friendships and trust relationships affect its beliefs. In the second part of the paper, we describe an alternative model that assumes two-way causality: relationships affect beliefs, but beliefs also affect relationships. This model adapts Hebb's learning algorithm for neuron firing and applies it to opinion dissemination on a signed network. The model is adequate to produce a wide variety of polarization behaviors, including belief polarization, relationship polarization, and social mitosis consistent with results in structural balance theory.

1 Hopfield Networks and Opinion Polarization

In this section, we consider a Hopfield network model for opinion polarization. The model features nodes with binary beliefs {0,1} and weighted relationship-links indicating trust or mistrust {+1, -1}. In the version of the model described in this section, we consider only one-way causality, whereby a node's relationships affect its beliefs. In the next section, we consider two-way feedback, whereby relationships affect beliefs but beliefs also affect relationships.

Hopfield networks were invented by John Hopfield in 1982 as a non-hierarchical artificial neural network model that was capable of remembering information. Hopfield networks can recover full memories from partial inputs or triggers, a functionality known as "content addressable memory."[1]

The standard Hopfield network model consists of a complete graph in which each node i can take on values $V_i = \{0,1\}$. A memory state is a particular configuration of node values, such as $\boldsymbol{V} = [0,1,1,0,1,1,...]$, corresponding to a bit-string message. An n node Hopfield network can therefore hold n bits of information.

[1] See Hopfield, 1982.

Edges are assigned weights of $w_{ij} = \{+1,-1\}$ according to the following algorithm[2]: given a set of s memory states and two nodes i and j let $w_{ij} = \sum_s (2V_i^s - 1)(2V_j^s - 1)$ but with $w_{ii} = 0$. Edges weights are static once the set of memory states is assigned, though additional memories can be added. Node weights are updated asynchronously and iteratively according to the following equations: $V_i = 1$ if $\sum_j w_{ij} V_j > U_i$ and $V_i = 0$ if $\sum_j w_{ij} V_j < U_i$ where U_i is a specified threshold. Once the edge weights have been trained, if a partial memory is fed in as the initial configuration of the network, e.g., $\boldsymbol{V}_0 = [0,0,1,0,0,1...]$, after a series of iterative node updatings, the network will return to the full memory, e.g., $\boldsymbol{V} = [0,1,1,0,1,1,...]$.

Hopfield proposed that network configurations could be thought of in terms of energy, defined as $E = -\frac{1}{2}\sum_{i,j} w_{ij} V_i V_j$. He showed that the network functions like a potential well, with states with higher energy transitioning to states with lower energy. Memories correspond to local energy minima. Partial memories can be thought of as higher energy states within the same potential well, or as states in the basin of attraction of a given stable memory state.

The Hopfield network has a plausible interpretation in the context of neuron function. The node values $V_i = \{0,1\}$ indicate whether a neuron is firing or not. The edge weights $w_{ij} = \{+1, -1\}$ correspond to synaptic relationships and indicate whether two neurons fire synchronously or anti-synchronously, consistent with Hebb's learning rule (i.e., "neurons that fire together, wire together"). The Hopfield updating algorithm in this context can then be understood as follows: a neuron receives input from each of the other neurons in the network; if it is synchronized with another neuron that is currently firing ($V_j = 1$), this increases the probability the neuron will fire, while if it is synchronized with another that is not firing ($V_j = 0$) this decreases the probability that it will fire. The effect of anti-synchronized neurons is the opposite. If the net input ($\sum_j w_{ij} V_j$) is above the neuron's firing threshold (U_i), then it fires ($V_i = 1$), otherwise it does not fire ($V_i = 0$). The synaptic wiring relationships in a Hopfield neural network encode one or more memories and provide a plausible mechanism whereby a set of neurons in the brain can recall information through synchronized firing.

Although the Hopfield model was originally developed in the context of neural networks, we propose that the mechanism and algorithms also have a plausible interpretation in the context of *social networks of belief and trust*. In this case, the node values $V_i = \{0,1\}$ indicate whether an individual believes a proposition or not. The edge weights $w_{ij} = \{+1,-1\}$ correspond to social relationships between individuals, indicating trust / mistrust or friendship / enmity. The Hopfield updating algorithm in this context can then be understood as follows: to determine whether to believe a proposition or not, I survey the neighbors in my social network; if I trust a network neighbor ($w_{ij} = +1$) and that neighbor believes the proposition ($V_j = 1$), then I am also more likely to believe it. If a trusted neighbor disbelieves the proposition ($V_j = 0$) then I am less likely to believe it. The effect of a mistrusted neighbor is the opposite. If the net commu-

[2] Edge weights may take on positive or negative integer values beyond {+1,-1} if *multiple* memories are stored in the network

nications I receive through my neighbor survey ($\sum_j w_{ij} V_j$) are above a certain belief threshold, then I will believe the proposition, otherwise not. This process might be described as a *social* Hebbian learning rule: just as "neurons that fire together, wire together," we may say that "agents that think together, drink together." [3]

The Hopfield mechanism has important implications for the study of belief and social influence. Just as the synaptic relations in a Hopfield neuronal network encode a set of stable attractor states corresponding to memories, *the trust relations in a Hopfield social network encode a set of stable attractor states corresponding to communal beliefs.*

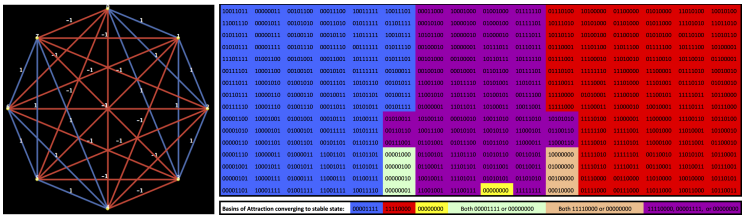


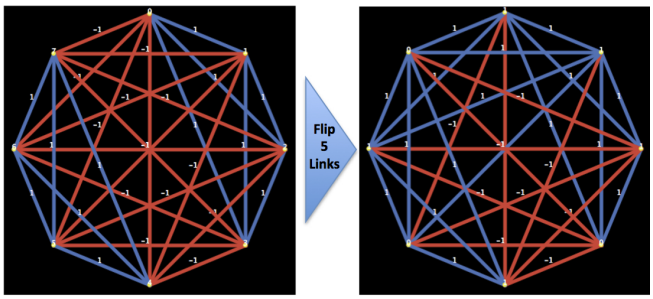
Fig. 1. Stable States and Basins of Attraction: example for an 8-node network with 2 factions. There are $2^8 = 256$ possible belief states (each of the 8 nodes can believe or disbelieve the proposition). Under the Hopfield updating algorithm, each these 256 starting states will converge to one or more of three stable attractor states ([1,1,1,1,0,0,0,0], [0,0,0,0,1,1,1,1], or [0,0,0,0,0,0,0,0]).

The state space of a Hopfield network is divided into a set of basins of attraction, each of which map to one or more stable states. These basins and stable states are determined by the edge weights. When a Hopfield neural network is given any of a set of partial memories, it reverts to a particular full memory. Similarly, when a Hopfield social network is placed in an unstable belief configuration, it will revert to a communal belief configuration pre-determined by the trust relations. The friendship or trust relations function as a form of *social memory.* Counterintuitively, this means that if we wish to understand what patterns of belief a community will converge to, it is the social relationships, not the current beliefs of the various individuals that matter most. The social relationships govern the dynamics of the system. The extant beliefs merely determine the initial conditions. In cases where there are only a few basins of attraction,

[3] One key difference between applying the Hopfield model to people vs. neurons is that people are likely to also give weight to their own pre-existing beliefs on a proposition rather than just summing over the trust-weighted beliefs of their neighbors. We can address this by modifying the node updating algorithm to $V_i^{t+1} = aV_i^t +$ (1-a)$\sum_j w_{ij} V_j^t$. With a = 0, individuals give no weight to their existing beliefs and decide only based on a survey of their neighbors. With a = 1, individuals have inflexible self-beliefs and ignore social influence entirely. (Note: In their implementation of a similar mechanism, Macy et al refer to this as the '"structural learning rate."')

these initial conditions have limited impact on the stable state to which the belief system ultimately converges.

These implications are particularly important to the study of *belief polarization*. Consider a network in which the edge weights are such that there are two cliques, with total friendship / trust within the groups (signified by all positive links) and total enmity / mistrust between the groups (corresponding to all negative links). In the context of the Hopfield mechanism, it makes sense to ask: "What are the stable belief states of this community? What "social memory" is encoded by their friendship and enmity ties?" For example, for an 8-node network with two cliques of 4, the most stable belief configurations are the polarized belief states [1,1,1,1,0,0,0,0] and [0,0,0,0,1,1,1,1]. No matter how varied the beliefs of the individuals composing this community initially are, beliefs are highly likely to polarize due to the architecture of friendship / trust ties.



Belief Stable States	Relative Frequency	Hopfield Energy
"11110000"	407	-3
"00001111"	404	-3
"00000000"	189	0

% runs ending in polarization = 81.1%

Belief Stable States	Relative Frequency	Hopfield Energy
"11001111"	690	-4.5
"11110000"	172	-3
"11110011"	115	-3.5
"00000000"	23	0

% runs ending in polarization: 17.2%

Fig. 2. Basins of Attraction for a Polarized vs. Modified Hopfield Network. The nodes are divided into 2 cliques initially, {0,1,2,3} and {4,5,6,7}. The stable states for the initial configuration are "11110000" (all nodes in clique 1 believe the proposition, all nodes in clique 2 disbelieve it), "00001111" (the inverse), or "00000000" (all nodes disbelieve the proposition and therefore agree with each other). A simulation is run 1000 times starting from a random state to determine how frequently the beliefs revert to each of the stable states. The percentage is initially 81.1% for the two polarized states. By flipping several link-relationships, the stable states change, such that polarized beliefs become significantly less likely (17.2%).

In the companion to this paper, "Measures of Polarization and Diversity," we propose a General Principle of Polarization: "the more effort required to bring people's attitudes into agreement, the greater the level of polarization." In a Hopfield context, we can operationalize this principle by asking the question: how many links do we need to change in order to *de-polarize* the stable states

of a network?[4] Figure 2 gives a simple example based on the 8-node network described above. If beliefs are assigned randomly to nodes in the network, there is a roughly 81.1% chance that these beliefs will converge to one of the 2 polarized stable states. By flipping just 5 of the trust / mistrust links, however, the stable states of the Hopfield network are altered and the probability of converging to a polarized state is reduced to only 17.2%.

This result is significant for the study of problems ranging from political deadlock to neighborhood de-segregation. It suggests that if we want to depolarize a community's beliefs or ideological commitments (say, Protestants and Catholics in Northern Ireland), trying to change individuals' beliefs is likely to be less effective in the long run than reorganizing the community's social ties. If we simply change individuals' beliefs, the network is likely to slide back into a polarized stable state eventually. If we change trust relationships, however, the polarized states can be eliminated and a different set of equilibria established.

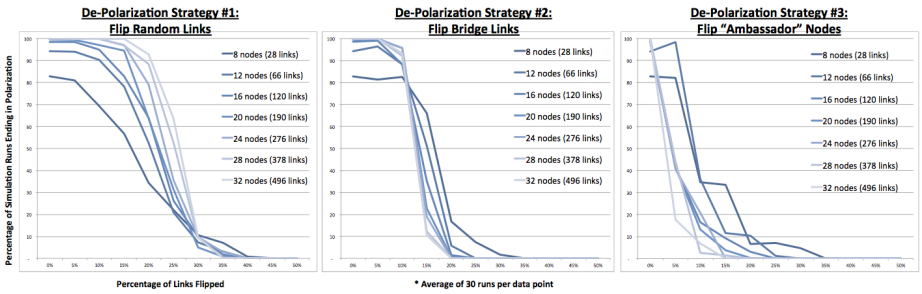


Fig. 3. % of simulation runs converging to polarization vs. % of links flipped.

Figure 3 offers a more systematic approach to the issue, showing the effect of three different de-polarization strategies on networks of different sizes.

The simplest strategy is to flip relationship-links at random: some friendships within a faction are changed to enmities and some enmities across factions are changed to friendships. This is reasonably effective for smaller networks, but becomes increasingly impractical for larger networks. Flipping 20% of the links in an 8-node network (that is, about 6 links) is adequate to significantly reduce the probability of belief polarization. But flipping 20% of the links in a 32-node network (that is, roughly 100 links) has barely any effect. It is not until we

[4] Rather than measuring the number of links required to depolarize a network, an alternative method of measuring the "effort required to bring people's attitudes into agreement" is the Hopfield Energy, $E = -\frac{1}{2}\sum_{i,j} w_{ij} V_i V_j$. We can measure the difference between the average Hopfield energy of the polarized stable states and that of the least costly de-polarized stable state. Letting F = effort, E_i = the energy of a depolarized stable state, and E_0 = the energy of the polarized stable state, we have the equation: $F = \min\{E_i - E_0\}$.

flip 30% (roughly 150 links) that such larger networks exhibit a phase change behavior and rapidly de-polarize.

A more effective de-polarization strategy for larger communities is, rather than flip relationship-links at random, to instead focus on what might be termed "bridge links," that is, enmity links that currently separate the two factions but could be turned to friendship or trust links bridging them. Practically speaking, this means having many individuals form friendships with members of the opposing faction. Using this strategy, a large network can be de-polarized by flipping just 15% of links.

An even more effective strategy is to focus on specific nodes. Instead of having many individuals form friendships with members of the opposing faction, we instead focus on a few individuals and get them to act as ambassadors or "welds," forming friendship and trust relationships with many individuals in the opposing faction. This more concentrated strategy proves the most effective for large networks: by flipping just 5% of the links in a community in this way, we can depolarize the stable states. It is also the most psychologically realistic: it is likely to be significantly easier to change the trust behaviors of a few targeted individuals across many relationships than many individuals each across a few relationships.

2 Hebbian Social Learning: Concept & Implementation

The Hopfield model described above assumes one-way causality between relationships and beliefs. In the remainder of this paper, we describe an alternative model that assumes two-way causality: relationships affect beliefs, but beliefs also affect relationships. The model adapts Hebb's learning algorithm for neuron firing and applies it to opinion dissemination on a signed network. As we will show below, the model is adequate to produce a variety of polarization behaviors, including opinion polarization, relationship polarization, and social mitosis (consistent with results in structural balance theory).

In the basic implementation of the model, each node in a complete graph is assigned a belief vector represented by a bit-string (e.g., [0,1,1,0,1]). Each entry in the vector can be thought of as the node's belief about a binary yes-no issue (e.g., "Should gay marriage be legalized?", "Should abortion be legalized?", etc.). Furthermore, each node is connected to each other node by a weighted, signed link. Positive weights indicate trust or friendship; negative weights indicate mistrust or enmity. Link weights are initialized randomly at + or - times a user-specified value, but can take on any continuous value in the range [-1, +1] as the model runs.[5]

The key assumption driving model dynamics is that friends prefer to be in the same belief state while enemies prefer to be in opposing belief states. If two friends have opposing beliefs, there is psychological pressure both to (1)

[5] Implementation of a similar mechanism can be found in Macy et al, 2003. Our approach is different insofar as it focuses on the typology and sequencing of polarization events.

update the social connection by shifting the link-weight from friendship towards enmity and (2) update individual beliefs so as to decrease the distance between belief vectors. Likewise, if two enemies find they have similar beliefs, there is psychological pressure both to: (1) update the social connection by shifting the link-weight from enmity towards friendship and (2) update individual beliefs so as to increase the distance between belief vectors. [6] [7]

- **Step 1) Belief Distance Calculation:** At each time-step, a random link is selected and the belief-distance between the end-nodes is calculated. The belief-distance b_{ij} between nodes i and j is defined as a modification of hamming distance. Let M be the number of features in the belief vector (equivalence to vector length). Let M^a_{ij} = the number of features where i and j agree; and M^d_{ij} = the number of features where i and j disagree. M = M^a_{ij} + M^d_{ij}. Then, $b_{ij} = \frac{M^a_{ij} - M^d_{ij}}{M} = \frac{M^a_{ij} - M^d_{ij}}{M^a_{ij} + M^d_{ij}}$. This form is chosen so that the belief-distances vary between -1 and +1.
- **Step 2) Social Updating:** The link-weight w_{ij} (indicating level of friendship vs. enmity) is updated by averaging it with the belief-distance: $w^{t+1}_{ij} = \frac{w^t_{ij} + b^t_{ij}}{2}$. If two nodes tend to agree on more issues than they disagree on, then the link-weight will be averaged with a positive number, etc; otherwise, negative, etc.
- **Step 3) Belief Updating:** We update the beliefs of the nodes with a random probability based on the absolute value of the link-weight. Thus, if the link-weight is +0.8, then there is an 80% chance that we will update the node beliefs. If the sign on the link-weight is positive, then we will flip one of the bits on one of the nodes so as to make the belief-vectors of the two nodes more similar. If the sign on the link-weight is negative, then we will flip one of the bits on one of the nodes so as to make the belief-vectors of the two nodes more different.

[6] There are a few key differences between this mechanism and the one-way Hopfield mechanism described above. First, the Hopfield algorithm updates nodes based on the sum-product of weights and neighbor beliefs ($\sum_j w_{ij} V_j$), whereas in the two-way model, we use a more general Hebbian-style learning algorithm based on probabilistic interactions. Second, link weights are dynamic in the two-way model and allowed to take on continuous values in the range [-1,1] rather than only static and discrete values {-1,+1} in the Hopfield model.

[7] It is worth commenting on the psychological plausibility and empirical research supporting this mechanism, which relies on four underlying assumptions: (1) *Homophily*: people who agree about beliefs tend to become friendlier and/or trust each other more (see McPherson et al, 2001), (2) *Xenophobia:* people who disagree about beliefs tend to become more hostile and/or mistrust each other more (see Schelling, 1971), (3) *Social influence and in-group conformity:* people's beliefs tend to adjust so as to conform with those they consider friends and/or those whom they trust (see Bernheim, 1994), (4) *Social differentiation and outgroup anti-conformity:* people's beliefs tend to adjust so as to oppose those they consider enemies and/or those whom they mistrust (see Simmel, 1908; Barth, 1969; and Rosenblum, 1986).

– **Repeat)** Steps 1-3 are repeated until either a pre-specified stop time (e.g., 10,000 ticks) or until key graph features reach steady-state.

3 Results for Binary Beliefs

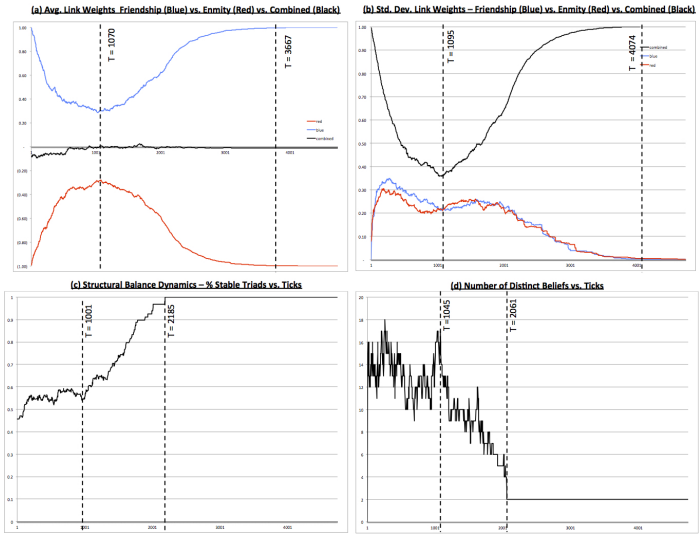


Fig. 4. Example Run (20 nodes, complete graph, 5 belief features, 2 belief traits)

Order of Convergence	Runs
Factions -> Beliefs -> Links	73
Factions -> Links -> Beliefs	0
Beliefs -> Factions-> Links	27
Beliefs -> Links -> Factions	0
Links -> Factions-> Beliefs	0
Links -> Beliefs-> Factions	0
Total Runs	100

Constants: 10 Nodes, Complete Graph, 5 belief features, 2 belief traits

Fig. 5. Table: Order of Convergence for Sample Runs

Figure 4 shows an example run of the model on a complete graph of 20 nodes. The time-trajectory of (a) average link weights, (b) standard deviation of link weights, (c) percentage of triads obeying structural balance stability rules,[8] and (d) total number of distinct beliefs are shown with significant times marked. Metrics for the link-weights are shown both in aggregate (black) and for friendship and enmity links separately (blue and red respectively).

At T = 0, the links and belief vectors are randomly assigned. Belief vectors for this run have five entries and are assigned as random bit-strings (e.g., [1,1,0,0,1]). Links are begun from weights of either +1 (maximum friendship) or -1 (maximum enmity).

[8] For a full explanation of structural balance theory (SBT), see Cartwright and Harary, 1956. In brief, SBT posits that on a signed network there are 2 stable triad configurations (+)(+)(+) and (+)(-)(-) and 2 unstable triad configurations (+)(+)(-) and (-)(-)(-). It can be shown that a complete graph satisfying SBT stability rules must be divided into two factions of friends with total enmity between them ("social mitosis").

Phase 1: In the first phase of the dynamics, lasting from T = 0 and to approximately T = 1000-1100, the number of distinct beliefs remains high and the number of stable triads low, gyrating around relatively constant averages. The main activity is related to the link-weights, which converge—both friendships and enmities become less extreme—as shown by the falling average and standard deviation.

Phase 2: beginning at roughly T = 1000-1100, a number of graph structural features begin to cohere simultaneously. Link-weights reach their maximum convergence and begin to extremize again. At the same time, the number of distinct beliefs begins to fall consistently and the number of structurally balanced triads begins to rise consistently, all of which are indicative of the formation of factions.

Phase 3: beginning at roughly T = 2100, structural balance and belief dynamics reach steady state. All triads satisfy structural balance rules and the number of beliefs reaches a minimum of 2. This is indicative of the fact that stable factions have formed. Consistent with SBT, the graph is now split into two mutually exclusive factions ("social mitosis") with total intra-group friendship and total inter-group enmity. The two factions also have distinct beliefs with all members of the same faction sharing the same belief vector. Not only are these belief vectors internally consistent, they are also diametrically opposed (one faction, consisting of 8 nodes, all believe [1,1,1,1,1] while the other, consisting of 12 nodes, all believe [0,0,0,0,0]).

Phase 4: from roughly T = 2100 to T = 4100, the link weights extremize until reaching maximum polarization. The factions and beliefs do not change during this period. All that occurs is that intra-group friendships become friendlier and inter-group enmities become more hostile.

Phase 5: from T = 4100 onwards, the graph and node beliefs are in steady state.

Though the mechanism is relatively simple, the model is adequate to generate a wide variety of polarization behaviors, including:

- 1. **Social Mitosis**: The graph splits into two mutually exclusive factions, with friendship between members of a faction and enmity between factions
- 2. **Link Polarization**: Friendship and enmity relationships are eventually extremized to +1 or -1 regardless of their starting values
- 3. **Belief Polarization**: The two social factions have perfectly uniform and perfectly opposed beliefs on each individual issue
- 4. **Belief Correlation**: The model produces correlation across multiple beliefs. The beliefs of the two groups are polarized on all issues, with no common ground. E.g., if all nodes in one faction converge to [1,0,1,1,0], then all nodes in the other facton will converge to [0,1,0,0,1]. The belief vectors will be perfect complements.

Moreover, the model produces a number of other noteworthy results.

First, even though the network is updating exclusively based on a Hebbs-style learning algorithm, the model reproduces the dynamics of classical structural balance theory. That is, even though we have updating of links and beliefs that is dyadic, it produces the same results as a model that is explicitly based on

triadic relations. The model efficiently stabilizes SBT-unstable triads locally and reproduces the familiar SBT result of social mitosis globally.

Second, different polarization events occur at different times in consistent sequences. As figure 3 shows, in the significant majority of model runs (73%), structural balance is achieved first with the formation of two separate factions, then beliefs polarize, and finally link weights polarize. The sequence aligns with intuition about group polarization processes. First, in-groups and out-groups form. Then, group members converge on a uniform set of beliefs. Finally, relationships extremize: in-group friendships become maximally friendly, out-group enmities become maximally hostile. Relationship gradations disappear and in-group / out-group become entirely black and white.

4 Results for Multi-Valued Beliefs

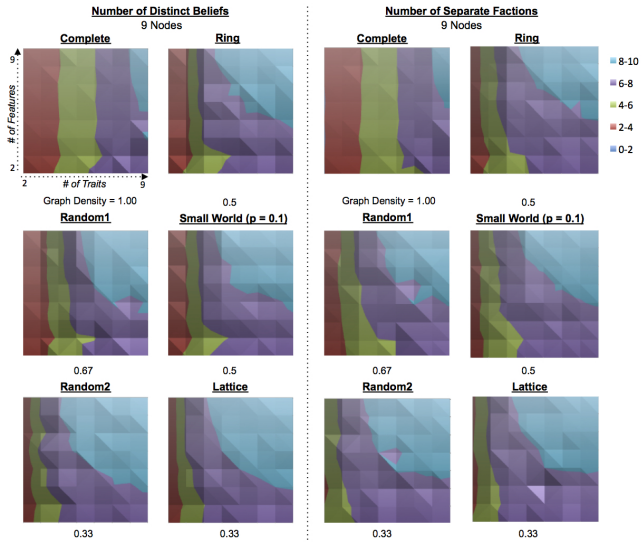


Fig. 6. Impact of network topology on # of distinct beliefs and # of separate factions in steady state. Average of 30 runs per parameter combination (11520 runs total).

In the basic version of the model, we assume (1) beliefs may take on only a value of 1 or 0, indicative of binary yes-no opinions, and (2) social relations occur on a complete graph. We extend the model by relaxing these two assumptions.

Node-beliefs are allowed to take on multiple discrete values (a.k.a., "traits"), indicative of categorical rather than binary beliefs, then the number of factions into which the network fractures may be greater than two. As with the binary case, within each faction, all nodes converge to maximal friendship and uniform beliefs, while nodes in opposing factions polarize, developing strong

enmity-relationships and opposing beliefs. However, while the Hebbs algorithm with binary traits produces dynamics consistent with classical structural balance, this is no longer the case if the number of traits is 3 or greater. In particular, the SBT rule that (-)(-)(-) is unstable—which guarantees at most 2 factions globally—is violated. Results do still agree with the three other local SBT rules however: (+)(+)(+) and (+)(-)(-) are stable, and (+)(+)(-) is unstable.

Figure 4 uses a series of heat maps to summarize the impact of (a) network topology, (b) number of belief features, and (c) number of belief traits on two key steady state characteristics: (1) the number of distinct belief vectors to which the graph converges and (2) the number of separate factions.

The first noticeable result is that *topology matters*. As graph density increases, the number of factions and the number of distinct beliefs increase: a complete graph (GD = 1.00) converges to fewer distinct beliefs and factions than a lattice (GD = 0.33) for all combinations of features and traits. This leads to a stylized fact: the less connected the community, the more diverse the set of beliefs and the more polarized the community is likely to be. Interestingly, the *number* of links in the graph is more important than the way those links are organized: e.g., the random and lattice graphs have the number of links (12 out of 36 possible = 0.33 graph density) but the links are organized differently, and yet the general pattern of belief fracturing is quite similar.

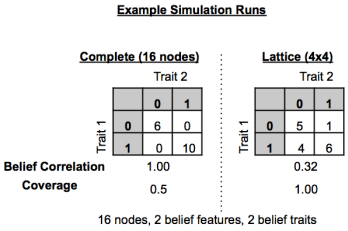
The second noticeable result is that as the number of *features* and *traits* increases (represented by the horizontal and vertical axes in figure 4, respectively), the number of distinct beliefs and factions. For a complete graph, the effect of the *traits* is linear: the number of factions always equals the number of traits. This is true independent of the number of features. For incomplete graphs, the number of distinct beliefs and factions in steady state generally exceeds the number of traits. Increasing the number of *features* has a more limited effect. For a complete graph, the number of features has no significant effect. For incomplete graphs, the number of features does matter. This is particularly true as the number of traits increases.

It is worth dwelling on why this is. If we have M features and N traits per feature, then there are, hypothetically, N^M possible belief vectors. For a complete graph, however, we find that there is perfect correlation between traits in steady state: thus if there are N traits, there are only N distinct belief vectors regardless of the number of features M. The "coverage"[9] for a complete graph will therefore be $\frac{N}{N^M} = N^{[1-M]}$. For an incomplete graph, there is not a perfect correlation between traits, and therefore there can hypothetically be as many as N^M distinct belief vectors.

Figure 5 shows a simple example run for the case that N = 2 and M = 2. For the complete graph, out of 4 possible belief vectors ([0,0] [0,1] [1,0] [1,1]), only 2 are realized (coverage = $\frac{2}{4} = 0.5$, which agrees with the equation $N^{[1-M]} = 2^{[1-2]} = \frac{1}{2}$) and they are diametrically opposed such that the correlation between

[9] In the companion to this paper, "Measures of Polarization and Diversity," we define "coverage" as the proportion of bins in the belief distribution that are occupied by at least one agent.

trait 1 and trait 2 is 1.00. For the lattice, there are nodes with each of the 4 possible belief vectors. Coverage $= \frac{4}{4} = 1.00$ and the correlation between trait 1 and trait 2 is only 0.32. The table in figure 5 shows a summary of results for various topologies assuming 16 nodes, 2 belief features, and 2 belief traits. The table shows that as graph density increases, belief coverage generally falls and the correlation between belief traits increases.



Graph Completeness vs. Belief Coverage and Correlation

Network Type	Graph Density	Belief Coverage *	Absolute Correlation *
random (15 links)	0.13	0.90	0.51
lattice	0.20	0.76	0.70
random (30 links)	0.25	0.69	0.79
small world (p = 0.1)	0.26	0.71	0.73
ring	0.27	0.72	0.77
random (45 links)	0.38	0.60	0.95
random (60 links)	0.50	0.54	0.98
random (75 links)	0.63	0.50	1.00
random (90 links)	0.75	0.51	1.00
random (105 links)	0.88	0.50	1.00
complete	1.00	0.50	1.00

* Average of 30 simulation runs

Fig. 7. Effect of graph completeness on coverage and correlation between belief traits

The third noticeable result in figure 4 is the relationship between the number of distinct beliefs and the number of factions. For a complete graph, these values are equal. That is, each faction has its own exclusive belief system that is not shared with any other faction. For incomplete graphs, the number of factions generally exceeds the number of distinct beliefs. That is, there are multiple factions that share the same beliefs. This is possible because the graph incompleteness means that these factions do not come into contact and merge but may be polarized identically relative to the same neighboring groups.

5 Conclusion and Discussion

In this paper, we have developed two models of belief polarization on signed social networks: (1) a Hopfield network model; and (2) a Hebbian social learning model. Both models are based on mechanisms originally developed in the context of neural networks but have plausible interpretations in the context of social networks of belief, friendship, and trust. The analogy between the domains has three parts: (1) social agents exhibit an independent "firing" behavior that is similar to neurons: they receive signals from network-neighbors and adjust their state vectors to conform or anti-conform, (2) social pathways such as friendships or trust operate like neuronal pathways: they transfer information about neighbor states but they do so non-neutrally, that is, the valence of the relationship affects how the information is interpreted; (3) belief agreement functions like synchronous neuron firing: correspondences in state space lead to reinforcement effects. The analogy is obviously broad and glosses over important differences, but the general relevance of the Hopfield and Hebbian mechanisms to social dynamics is provocative, suggesting that Hebbian learning as an organizing principle may apply at multiple levels of systems organization.

References

1. Axelrod R (1997) The dissemination of culture: A model with local convergence and global polarization. Journal of Conflict Resolution 41: 203226.
2. Barth F (ed.) (1969) Ethnic Groups and Boundaries: The Social Organization of Culture Difference. Boston: Little, Brown.
3. Aaron Bramson, Patrick Grim, Daniel J. Singer, Steven Fisher, Graham Sack, William Berger, and Carissa Flocken. "Measures of Polarization and Diversity." Computational Social Science Society of the Americas (CSSSA) 2013 Proceedings. Sante Fe, NM August 22-25, 2013.
4. Cartwright, D Harary, F. (1956). Structural Balance: A Generalization of Heiders Theory. Psych. Review. V. 63, No. 5.
5. Bernheim D (1994). A theory of conformity. Journal of Political Economy 102(5): 841877.
6. Hebb, D. (1949). "The Organization of Behavior." Wiley, New York.
7. Hopfield, John (1982). "Neural networks and physical systems with emergent collective computational abilities." Proceedings of the National Academy of Sciences. USA. Vol. 79, pp. 2554-2558, April.
8. Macy MW, Kitts JA, Flache A, and Benard S (2003) Polarization in dynamic net- works: A Hopfield model of emergent structure. In: Breiger R, Carley K, and Pattison P (eds) Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers. Washington, DC: The National Academies Press, 162173.
9. McPherson M, Smith-Lovin L, and Cook JM (2001) Birds of a feather: Homophily in social networks. Annual Review of Sociology 27: 415444.
10. Rosenbaum, M.E. 1986. The repulsion hypothesis: On the non-development of relationships. Jour of Pers Soc Psy 51: 1156-1166.
11. Schelling TC (1971) Dynamic models of segregation. Journal of Mathematical Sociology 1: 143186.
12. Simmel G (1955) Conflict and the Web of Group Affiliations. New York: The Free Press.

6 ODD

6.1 Purpose

The purpose of this model is explore the application of the Hebb's learning algorithm for neuron firing to signed social networks and determine its potential relevance as a plausible mechanism for generating belief polarization.

6.2 Design Concepts

The key concept behind the model is that friends prefer to be in the same belief state while enemies prefer to be in opposing belief states. If two friends have opposing beliefs, there is psychological pressure both to (1) update their social connection by shifting the link-weight from friendship towards enmity and (2) update their beliefs so as to decrease the hamming distance between their belief vectors. Likewise, if two enemies have similar beliefs, there is psychological pressure both to: (1) update their social connection by shifting the link-weight from enmity towards friendship and (2) update their beliefs so as to increase the hamming distance between our belief vectors.

The mechanism draws upon bodies of psychological and sociological literature related to *homophily* (e.g., McPherson et al, 2001), *xenophobia* (e.g., Schelling, 1971), *social influence and in-group conformity* (e.g., Bernheim, 1994), and *social differentiation and outgroup anti-conformity* (e.g., Simmel, 1908; Barth, 1969; and Rosenblum, 1986).

In addition, our design is influence by several model paradigms:

1. Hebbian Learning: The Hebbian learning rule in a neural context is crudely summarized by the heuristic "cells that fire together, wire together"—meaning that the strength of the pathways between two neurons in the brain will adjust to match the frequency with which those neurons fire together. We are concerned with the application of the Hebbian learning rule to a social context, which might be crudely summarized by the heuristic "agents that think together, drink together." The analogy has three parts:

- (i) social agents exhibit an independent "firing" behavior that is similar to neurons: they receive signals from network-neighbors and adjust their state vectors to conform or anti-conform
- (ii) social pathways such as friendships or trust operate like neuronal pathways: they transfer information about neighbor states but they do so non-neutrally, that is, the valence of the relationship affects how the information is interpreted
- (iii) belief agreement functions like synchronous neuron firing: correspondences in state space lead to reinforcement effects.

2. Axelrod's Dissemination of Culture Model: In "Dissemination of Culture," Axelrod proposed a rule for local cultural convergence whereby each patch in a grid has a cultural vector consisting of M features and N possible traits per

feature (e.g., if M = 5 and N = 3, one possible cultural vector could be [1 0 2 1 2]). Two neighbors interact with a probability equal to the percentage of cultural features that they share. Thus, if A has vector [1 0 2 1 2] and B has a vector [0 2 2 0 2], there is 40% chance of interaction. If two nodes interact, then they become more similar, altering an additional feature to match. Axelrod shows that this local convergence rule is adequate to produce global polarization dynamics.

In addition to re-contextualizing Hebbian Learning, our model is also intended as an extension of the Axelrod model to (1) signed social networks and (2) topologies beyond a grid / lattice.

In the Axelrod model, if two nodes share a trait, then there is a positive probability that they will interact and if they interact, they will become more similar, leading to a positive feedback effect. Axelrod's model has no analogous effect related to difference and further differentiation. In our model, not only may existing similarities lead to convergence but differences in traits may lead to further differentiation.

Axelrod's model relies on two psycho-social processes:

- (1) *Homophily:* people who agree about beliefs tend to interact more
- (2) *Conformity and Social Influence:* people's beliefs tend to adjust so as to conform with those with whom they interact

Our model considers two additional effects:

- (3) *Heterophobia:* people who disagree about beliefs tend to become more hostile and/or mistrust each other more
- (4) *Anti-Conformity and Social Differentiation:* people's beliefs tend to adjust so as to oppose those they consider enemies and/or those whom they mistrust.

We use signed networks to capture relationship valences indicating friendship / enmity or trust / mistrust.

3. Structural Balance Theory: Both the Hebbian and Axelrod models are concerned with dyadic interactions between two nodes. As a design principle, we are also interested in triadic relationships, inspired by structural balance theory.

Structural balance theory, also known as social balance theory (SBT), was originated in the mid-1940s by Fritz Heider, who studied patterns of belief coherence in individual psychology. In the mid-1950s, Cartwright and Harary generalized Heider's theory of coherence and applied it to social relations, representing stable and unstable configurations with basic graph theory.

Consider a set of nodes representing, for example, people or countries. Each node may be joined to each other node by an edge, which represents their relationship. If two nodes are joined, they are either (1) friends or (2) enemies. The fundamental unit of analysis in SBT is a triad of three mutually linked nodes. A triad is considered unstable if there is social pressure to change one of the relationship links. It is considered stable if there is no social pressure to change.

Let (+) represent friendship and (-) represent enmity. There are several possible configurations:

- 1. (+)(+)(+): If all 3 nodes are friends / allies, the triad is considered stable.
- 2. (-)(-)(-): If all 3 nodes are enemies, the triad is unstable, since two nodes have an incentive to ally against the third (thereby becoming friends with each other).
- 3. (+)(+)(-) or (+)(-)(+) or (-)(+)(+): If one node is friends with two that are enemies with one another, it will be pressured to pick a side, and therefore the triad is unstable.
- 4. (+)(-)(-) or (-)(+)(-) or (-)(-)(+): If two nodes are friends with each other and both are enemies against a third, the triad is stable.

The rule for stability can be summarized as follows: a triad is stable if the multiplicative product of the signs is positive. The stability of the various triads conforms to the following simplified social principles: (1) my friend's friend is my friend; (2) my friend's enemy is my enemy; (3) my enemy's friend is my enemy; (4) my enemy's enemy is my friend.

The stability of three mutually connected nodes is easy enough to evaluate, but the complexity increases as nodes are added to create larger graphs with many interdependent triads. Nevertheless, global patterns emerge from the local interactions. One such pattern is *social mitosis*: it can be proven that there are only two ways for a complete graph, i.e., one with no missing edges, to be structurally balanced: (1) everyone is friends (universal harmony); or (2) there are two factions of friends with total enmity between them (bi-polar factions).

Although the mechanism driving our model dynamics is dyadic, we measure and track triadic properties, particularly SBT stability and the number of factions that form in equilibrium. Likewise, we are interested in bringing state-vector mechanisms to bear on SBT—the classical version of which is concerned only with node relationship-links and does not include any node attributes, such as beliefs.

6.3 Entities

The model contains three types of entities: (1) nodes; (2) links; (3) triads.

Nodes: Each node in the network has a belief vector represented by a bit-string (e.g. [0 1 1 0 1]). Each entry in the vector can be thought of as the nodes belief about a binary yes-no issue (Should gay marriage be legalized?, Should abortion be legalized?, etc.). In addition, nodes have attributes that track their degree and the triads to which they belong.

Links: Each node is connected to each other node by a weighted, signed link. Positive weights indicate friendship; negative weights indicate enmity. Link weights are initialized randomly at + or - times a user-specified value, but can take on any continuous value in the range [-1, +1] as the model runs.

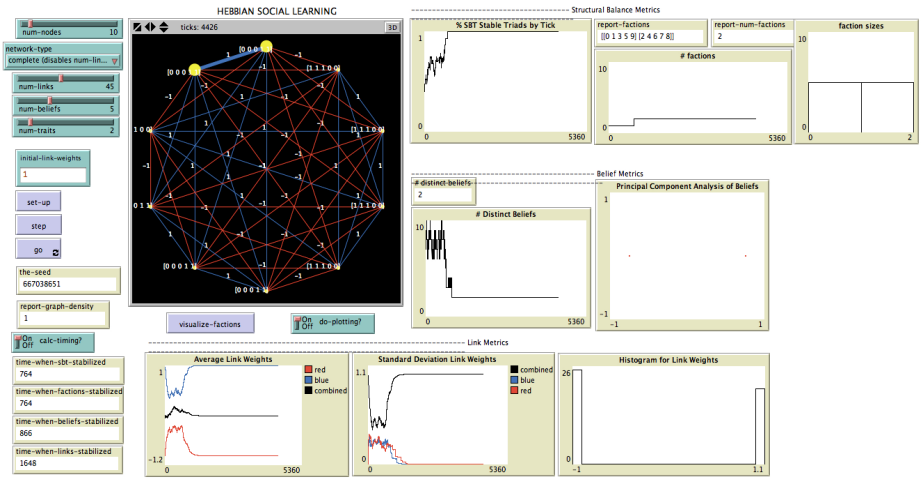


Fig. 8. Snapshot of Netlogo Model

Triads: Triads are sets of three mutually-connected nodes. Triads have several attributes: (1) a list of the nodes belonging to that triad; (2) a list of three links belonging to that triad; (3) a boolean variable indicating whether the triad is stable or not according to the rules of structural balance theory. SBT posits that on a signed network there are 2 stable triad configurations (+)(+)(+) and (+)(-)(-) and 2 unstable triad configurations (+)(+)(-) and (-)(-)(-) (for a full explanation of structural balance theory (SBT), see Cartwright and Harary, 1956.)

6.4 Set-up Procedures and Input Parameters

At set-up, the user specifies the following:

- the number of nodes
- the network topology: complete, lattice, ring, small world (with rewiring probability of 0.1), random
- the number of links: this input is disabled for topologies where the number of links is determined by the number of nodes, for example, complete, lattice, ring, and small world networks.
- the length of the belief vector (after Axelrod 1997, we will refer to this as the number of belief "features")
- the range of discrete values allowed per vector entry (after Axelrod 1997, we will refer to this as the number of belief "traits")
- an initial link value w_0 (all links will randomly assigned $+w_0$ or $-w_0$)

In the snapshot of the Netlogo model above, these inputs are shown as the sliders and dropdown menus in the upper left corner of the model interface.

A signed network is then constructed satisfying the user-specified criteria.

6.5 Go Procedures

At each time step, the following procedures are executed:

- **Step 1) Belief Distance Calculation:** At each time-step, a random link is selected and the belief-distance between the end-nodes is calculated. The belief-distance b_{ij} between nodes i and j is defined as a modification of hamming distance. Let M be the number of features in the belief vector (equivalence to vector length). Let M^a_{ij} = the number of features where i and j agree; and M^d_{ij} = the number of features where i and j disagree. M = M^a_{ij} + M^d_{ij}. Then, $b_{ij} = \frac{M^a_{ij} - M^d_{ij}}{M} = \frac{M^a_{ij} - M^d_{ij}}{M^a_{ij} + M^d_{ij}}$. This form is chosen so that the belief-distances vary between -1 and +1.
- **Step 2) Social Updating:** The link-weight w_{ij} (indicating level of friendship vs. enmity) is updated by averaging it with the belief-distance: $w^{t+1}_{ij} = \frac{w^t_{ij} + b^t_{ij}}{2}$. If two nodes tend to agree on more issues than they disagree on, then the link-weight will be averaged with a positive number, etc; otherwise, negative, etc.
- **Step 3) Belief Updating:** We update the beliefs of the nodes with a random probability based on the absolute value of the link-weight. Thus, if the link-weight is +0.8, then there is an 80% chance that we will update the node beliefs. If the sign on the link-weight is positive, then we will flip one of the bits on one of the nodes so as to make the belief-vectors of the two nodes more similar. If the sign on the link-weight is negative, then we will flip one of the bits on one of the nodes so as to make the belief-vectors of the two nodes more different.
- **Repeat)** Steps 1-3 are repeated until either a pre-specified stop time (e.g., 10,000 ticks) or until key graph features reach steady-state.

Note that steps #2 and #3 are done separately (that is, belief updating depends on the original link-weight, not the updated link-weight). The net effect should be at each step to make the link-weights agree a little more with the beliefs and make the beliefs agree a little more with the link-weights.

Once the network has reached a steady-state, the user can press the VISUALIZE-FACTIONS button, which will kill all of the enmity links, separating the network into multiple cliques if it is polarized. The size of the factions are displayed in output boxes.

6.6 Submodels

Set-Up Procedures

set-up: the set-up procedure performs the following sequence of actions: (1) reset all model variables and clear plots, (2) reset the random number generator, (3) call check-valid-inputs to ensure that all user specific parameters are in permitted ranges, (4) initialize-globals, (5) construct network given user-specified

topology, number of nodes, and number of links, (6) initialize nodes, (7) initialize links.

check-valid-inputs(): This procedure checks that the number of links is not too large relative to the number of nodes. The maximum number of permissible links is nC2, where n is the number of nodes. If the number of user specified links exceeds the number allowed, the procedure throws an error and stop the program

initialize-globals(): this procedure initializes the temporal variables used to determine whether or not the graph has reached steady-state yet. Once the graph reaches steady-state, these variables (time-when-links-stabilized, time-when-sbt-stabilized, time-when-beliefs-stabilized, and time-when-factions-stabilized) are populated with relevant times. Until then, they are populated with "n/a"

construct-network: this procedure constructs a network with the user-specified topology. The options are (1) complete network: all nodes are directly linked to all other nodes; (2) lattice: nodes are laid out in a grid and are directly connected to their Moore-4 neighbors; (3) ring: nodes are laid out in a circle and are directed connected to their immediate left, right, left-left, right-right neighbors; (4) small world network: a ring network with random rewiring probability of 10%; (5) uniform network: a network with a uniform degree distribution; (6) maximum inequality: a network constructed so as to maximize the standard deviation in node degree given the specified number of nodes and links.

initialize-nodes(): this procedure (1) creates the user-specified number of nodes indicated by the slider num-nodes on the interface; (2) colors, sizes, and locates them spatially; (3) assigns each node a random "belief" vector that has the user-specified number of features and number of traits. For example, if the there are 5 features and 2 traits, a node might be randomly assigned the initial belief vector [1 0 1 1 0]; (4) calculates the node's degree and the set of complete triads to which it belongs.

initialize-links: (1) assigns each link an initial value $+w_0$ or $-w_0$, where w_0 is a user-specified initial-link-weight; (2) colors positive links blue and negative links red.

Go Procedures

go: the go procedure is performed at each tick. the procedure first checks to see whether the graph is in steady-state by calling evaluate-belief-stability, evaluate-faction-stability, evaluate-sbt-stability, and evaluate-link-polarization. If the graph has not already reached steady-state, then the go procedure performs the following sequence of steps: (1) randomly select one link; (2) identify the end-nodes of that link; (3) call update-node-beliefs(node1, node2) on the end-nodes of the link; (4) call update-link-weight() on the given link; (5) update the triad stability

for all triads containing that link; (6) call the plotting procedures.

report-belief-dist(): The belief-distance b_{ij} between nodes i and j is defined as a modification of hamming distance. Let M be the number of features in the belief vector. Let M_{ij}^a = the number of features where i and j agree; and M_{ij}^d = the number of features where i and j disagree. M = $M_{ij}^a + M_{ij}^d$. Then, $b_{ij} = \frac{M_{ij}^a - M_{ij}^d}{M}$ = $\frac{M_{ij}^a - M_{ij}^d}{M_{ij}^a + M_{ij}^d}$. This form is chosen so that the belief-distances vary between -1 and +1.

make-more-different(node1, node2): this procedure locates an entry in the two nodes' bit-strings that is the same (if such a bitexists) and then changes it for one node so that nodes values for that entry differ.

make-more-similar(node1, node2): this procedure locates an entry in the two nodes' bit-strings that differs (if such a difference exists) and then changes it for one node so that nodes values for that entry match.

report-all-different-bits(bit-string1, bit-string2): reports all bit positions where two bit-strings differ.

report-all-same-bits(bit-string1, bit-string2): reports all bit positions where two bit-strings match.

report is-disconnected-graph(): reports if the graph is contiguous (forms one giant component such that every node be reached by some path from every other node) or disconnected (the graph contains isolates or disconnected subcomponents).

friends-of(node): this procedure returns a list of nodes that have positively-weighted links to the given node.

report-factions(): this procedure returns a list of all factions in the network: it has the form of a list of lists in which each entry contains a set of nodes that form a faction. Factions are determined by randomly selecting nodes, then building a list of their friends, the friends of their friends, and so on. If there is a path along friendship-links from one node to another, then they are in the same faction. If there is not, they are in separate factions.

visualize-factions(): this procedure visualize the separate factions in the network. It does so by removing all of the enmity links in the network. If two sets of nodes are connected by only enmity links, then they are separate factions.

report-num-factions(): reports the number of factions in the network.

report-faction-sizes(): reports the relative sizes of the factions in the network.

report-triad-list(): this procedure scans the graph for complete triads of nodes. It does so by checking whether link-neighbors of a node are link-neighbors with each other. These triad lists are used for structural balance stability calculations.

evaluate-belief-stability(): returns a boolean indicating whether the belief vectors have stabilized. This is done by determining whether the set of beliefs held by nodes has changed in the last 1000 ticks.
report-num-distinct-beliefs(): reports the number of distinct belief vectors held by nodes in the network.

evaluate-faction-stability(): returns a boolean indicating whether the network factions have stabilized. This is done by determining whether the set of factions has changed in the last 1000 ticks.

evaluate-link-polarization(): returns a boolean indicating whether the links weights in the network have stabilized. This is done by determining whether the any link weight has changed in the last 1000 ticks.

report-triad-stability(input-triad): this procedure returns a boolean indicating whether a given triad is stable according to structural balance rules. The triad is stable if the links valences are (+)(+)(+) or any cyclical permutation of (+)(-)(-). The triad is unstable if the links valences are (-)(-)(-) or any cyclical permutation of (+)(+)(-).

evaluate-sbt-stability(): this procedure returns a boolean indicating whether there exists at least one unstable triad in the network based on structural balance theory rules.

report-graph-density(): calculates the graph density of the network by dividing the actual number of links in the network by the total number of possible links (= nC2, where n = number of nodes).

update-node-beliefs(node1, node2): This procedure does the following: (1) retrieve the weight w_{ij} of the link between two nodes; (2) if w_{ij} >0, then the nodes are friends and with probability = abs(w_{ij}), we make their beliefs more similar by calling make-more-similar(node1, node2); (3) if w_{ij} <0, then the nodes are enemies and with probability = abs(w_{ij}), we make their beliefs more different by calling make-more-different(node1, node2).

update-link-weight(given-link): update the weight on the link by averaging it with the belief-distance between its end-nodes.

report-belief-correlation(): creates a sequence of vectors consisting of all the traits

values for feature 1 vs. feature 2 vs. feature 3,.. across all nodes in the network, then finds the correlation between these trait-vectors. The values show how correlated the traits are to one another in the beliefs of the nodes.

fact(n): this procedure returns n!

comb(n,r): this procedure calculates nCr

Plotting Procedures:

do-num-factions-plot(): plots the number of factions in the graph vs. ticks

do-num-beliefs-plot(): plots the number of distinct beliefs in the graph vs. ticks

do-sbt-plot(): plots the percentage of triads in the network that satisfy structural balance stability rules vs. ticks

do-link-weight-average-plot(): calculates the average absolute value link-weight for three groups: (1) friendship links (those with w_{ij} >0) (2) enmity links (those with w_{ij} <0) (3) all links. Averages are plotted vs. ticks

do-link-weight-stdev-plot(): calculates the standard deviation of link-weights for three groups: (1) friendship links (those with w_{ij} >0) (2) enmity links (those with w_{ij} <0) (3) all links. Standard deviations are plotted vs. ticks

do-link-weight-histogram(): bins the link-weights and plots them as a histogram

do-faction-size-bar-plot(): plots the relative sizes of the factions

report-belief-covariance(node1,node2): reports the covariance between the belief vectors of two nodes.

report-belief-covariance-matrix(): constructs a covariance matrix by calculating the belief-covariance between each pair of nodes in the network.

report-principal-components(): performs principal component analysis on the belief vectors of the nodes in the matrix by (1) calculating the eigenvectors of the belief-covariance matrix (2) selecting the eigenvectors with the two highest eigenvalues

do-pca-plot(): plots all of the node belief vectors along the two principal component eigenvectors. this is a means of visualizing the belief space and seeing whether the belief have converged.