UNIVERSITY OF AMSTERDAM

MASTER THESIS

THIS THESIS IS A WORK IN PROGRESS

Modelling Meta-Agreement through an Agent-Based Model

Author:
Amir Sahrani

Examiner:
Dr. Fernando P. Santos
Supervisor:
Prof. Dr. Ulle Endriss
Assessor:
Dr. Davide Grossi

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computational Science

in the

Computational Science Lab
Informatics Institute

May 5, 2025

Declaration of Authorship

I, Amir Sahrani, declare that this thesis, entitled 'Modelling Meta-Agreement through an Agent-Based Model' and the work presented in it are my own. I confirm that:

- □ This work was done wholly or mainly while in candidature for a research degree at the University of Amsterdam.
- □ Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- □ Where I have consulted the published work of others, this is always clearly attributed.
- □ Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- □ I have acknowledged all main sources of help.
- □ Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Your signature

Date: May 5, 2025

“The majority, standing in for the people, wills everything and therefore wills nothing”

Joshua Cohen

Abstract

Include your abstract here Abstracts must include sufficient information for reviewers to judge the nature and significance of the topic, the adequacy of the investigative strategy, the nature of the results, and the conclusions. The abstract should summarize the substantive results of the work and not merely list topics to be discussed.

Length 200–400 words.

Acknowledgements

Thank the people that have helped, supervisors family etc.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

SYMBOLS

N	The set of all voters
X	The set of all alternatives
$\succ$	A preference relationship
$\mathcal{D}$	A domain of possible profiles
D	A deterministic deliberative procedure
DB	A deliberative procedure with biased voter
$\mathcal{L}(A)$	Set of all possible preference order over A
R	Set of a preference relations over all candidates
$\boldsymbol{R}$	Set of preferences of all voters
f	A function mapping a strict profile to a candidate
$\triangleleft$	A geometric order over candidates
Ψ	Vector of all policies
ψ	An instance of a policy
S	Vector of support for each policy
Σ	$\lvert A\rvert \times \lvert\Psi\rvert$ matrix estimating support of policies for each alternative

CHAPTER 1

METHODS

We proceed with the methods used to replicate the paper by **?**], as well as the experimental setup of our own model. Links to the data used for these experiments can be found in **??**[1]. The programs are implemented using `OCaml`, and `Python`.

1.1 Replication

We implement the model as described in **??**, agents are only allowed strict preferences over all candidates. All experiments are done with 3 alternatives, and 51 voters. The number of voters is specifically chosen to be an odd number, as this prevents perfect ties between alternatives. We measure all evaluations relating to strict preferences, as reported by **?**], in addition to those we also measure the number of Condorcet winners.

1.2 Experiments

We aim to replicate the findings by the AMERICA IN ONE ROOM experiments [1], to this end we use two models. Firstly we use the adapted DeGroot model as laid out in **??**[2], then we extend these results using our Agent Based model. The original experiment had a control group as well as the experimental group. To model the control group, we map all the voters onto a graph, we explain this mapping in the next section, as well as its computational difficulties. The experimental group is simply modelled as a densely connected network. The trust matrix is generated based on the graph the voters

[1] Some references will be broken since I have simply commented out all irrelevant sections

[2] This section is excluded, but in summary it is essentially a stochastic matrix, where the entries are decided based on a graph and some parameters. Using this trust matrix we take power of this matrix, and at different powers (i.e. time steps) we left multiply the support matrix and estimated support matrix to get the "outcome" of the deliberation at that time.

are embedded in, as such they can only trust voters they have a connection to in the underlying graph. The distribution of this trust we control though 3 methods. Firstly, and most simply, we give all voters a bias. This bias reflects how much of their trust they place on themselves. For example a bias of 1 represents them placing equal trust on themselves as all other voters combined, the actual weight on the self loop is calculated as the sum of all incoming edges multiplied by the bias. Secondly, we have knowledge based trust, in which a voter trusts voter j more if voter j is more knowledgeable. We get the knowledge scores from the AMERICA IN ONE ROOM dataset by taking the proportion of knowledge questions they answered correctly. The interpretation is that more knowledgeable voters would be more persuasive and thus be more influential on other voters' opinions. Thirdly, we have credibility based trust, where the trust a voter places on another voter is proportional to the number of outgoing edges that second voter has. This method becomes equivalent to placing uniform trust in all voters when all voters are situated in a fully connected graph. If we do not use credibility or knowledge based trust, we call this uniform trust, meaning that they treat all neighbors the same, importantly this does not imply any specific bias value.

1.2.1 Voter Mapping

In order to simulate realistic information flow through the control group, we aim to use a natural graph structure, as well as a natural mapping from voters to nodes. Firstly, in order to generate the graph, a starting graph is taken, namely the graph of academic citations, and the TIES [2] algorithm is then used to sample exactly n nodes from this graph. The TIES Algorithm first samples an edge, and adds both the source and target node to the new graph, these stage is called the sampling stage. After the desired number of nodes has been reached, we proceed to the induction step, during which all the edges that exist between the sampled nodes in the original graph are added to the new graph. This algorithm allows for the use of large, natural graphs, by scaling them down to the number of nodes desired.

Once the proper graph is generated, we calculate the pairwise shortest paths between all nodes, as well as the distance in voter opinions. We then try to find a bijection between the voters and the nodes such that the difference between the shortest path and the opinion distance is minimized.

We now proceed to show that mapping voters to a graph as just described is NP-Hard, we call this problem Distance based Voter Mapping and prove two statements regarding its complexity.

Theorem 1.1. Distance based Voter mapping is NP-Hard, when using the ℓ_2-norm

Proof. The proof follows from a reduction to the Quadratic Assignment Problem (QAP).

Let A be the matrix of pairwise distances between voters, and let B be the matrix of shortest-path distances in the graph G. Mapping the voters to nodes in the graph requires finding a bijection f that minimizes the following objective:

$$\sqrt{\sum_i \sum_j \left(A_{i,j} - B_{f(i),f(j)}\right)^2}.$$

Since the square root is a strictly increasing function, minimizing the expression above is equivalent to minimizing the sum inside:

$$\sum_{i,j} (A_{i,j} - B_{f(i),f(j)})^2.$$

Expanding the square gives:

$$\sum_{i,j} A_{i,j}^2 - 2A_{i,j}B_{f(i),f(j)} + B_{f(i),f(j)}^2.$$

The terms $\sum A_{i,j}^2$ and $\sum B_{f(i),f(j)}^2$ are independent of f (the former is fixed, the latter is a permutation of a fixed matrix), so the optimization reduces to:

$$\max_f \sum_{i,j} A_{i,j}B_{f(i),f(j)},$$

which is the standard form of the Quadratic Assignment Problem.

Finally, we note that the only restriction on A is that it arises from embedding voters in a Euclidean space. However, even under this constraint, QAP remains NP-hard [3]. □

Corollary 1.2. Distance Based Voter mapping is NP-Hard, when using the ℓ_1-norm

Proof. First assume that matrices A and B, representing the same matrices as before, are now binary matrices. Under this assumption the ℓ_1- and ℓ_2-norms are identical. Under this –strong– restriction, we have reduced the problem to 0-1-QAP, which is NP-Hard [4]. □

One concern with Theorem 1.1, is that the data might contain certain patterns that we might be able to exploit, though the problem statement does not require such patterns to exist, if they do, such patterns seem unlikely to be of much help. Take for example the case in which all voters hold one of 2 opinions, thus we can split them into two groups

of sizes n_1, n_2, then the mapping algorithm effectively requires finding a partition in the graph, that results in two sub-graphs with exactly n_1 and n_2 nodes each. This is now the size-constrained graph partitioning problem, which is NP-Hard. Thus, given that even under such a strong assumption the problem remains computationally intractable, we suspect that patterns in the data are unlikely to allow for better exact solutions. Despite these negative results, we enlist the help of QAP-solver [5] to find solutions, using the Fast Approximate QAP Algorithm [6]. We find the solver does not consistently find better solutions than random assignment. Given the number of simulation ran, it is infeasible to attempt to refine solutions to consistently be better than random solutions.

1.2.2 DeGroot extension

The first experiments we perform concern the DeGroot model. These experiments consist of two parts. Firstly we search the parameter space to identify parameters that best replicate the data, using Bayesian Parameter Estimation. For this we use data from the AMERICA IN ONE ROOM experiment as described in **??**. Though this data does not provide full preference rankings over the candidates, it does provide data on voters' opinions on 6 different topics of political discussion, such as climate change and immigration. Using these opinions, we are able to generate potential candidates, this is done either by selecting a voter and creating a candidate with identical opinions, or by pooling 10 voters[3] and creating an average of their opinions. Using these simulated candidates we are able to create preference rankings using the ℓ_1-norm. To model the difference between the deliberation and control group we change the topology of the graphs voters in the respective groups are situated in. As mentioned before, the deliberation groups will be embedded in a fully connected graph, while the control groups will be placed on the graph of academic citations in physics [7][4], this graph is small enough to allow sampling of the graph for each simulation. Since the original data provides group numbers for candidates who participated in the deliberation, we also experiment with replicating these groups as opposed to randomly grouping voters together.

We measure whether the final profiles are cyclic, whether they have a Condorcet winner, home many unique profiles there are, and their proximity to being single-peaked. Proximity to single-peakedness is measured in two ways. When the simulation size allows for it, we measure the proximity in terms of the number of voters that would need to be removed for the full profile to become single-peaked. This particular notion is NP-complete [8], though it allows for a 2-approximation, we use the method based on an

[3] This is arbitrary, and it might be good to look into this, but in my opinion this is low priority for now. It might also be useful to keep the candidates constant over the course of an experiment

[4] It might also be useful to compare to different graphs, but for now it seems okay to mention the graph's statistics and how they compare to other social networks.

Parameter	Description
`Number of Voters`	The number of voters in the simulation, representing either the deliberation group, or the control population.
`Number of Candidates`	The number of candidates to be voted on.
`Candidate Generator`	The way the candidates are generated. Either a random voter is selected for each candidate, or 10 random voters get averaged into one candidate.
`Bias`	The bias all voters have towards their own opinion.
`Time steps`	The number of deliberation "steps" the voters undergo.
`Group`	Use the original groups.
`Credibility`	Distribute trust based on credibility.
`Knowledge`	distribute trust based on knowledge.

TABLE 1.1: The parameters of the DeGroot learning based model, as well as their descriptions

ILP solver, as implemented in PrefTool [9]. The other notion of proximity is the proximity in terms of the number of candidates that need to be removed for the profile to become single-peaked. This can be done in $O(|V| \cdot |C|^3)$[10], though the implementation we use is that of the `PrefTools` library [9], which implements a slower $O(|V| \cdot |C|^5)$ algorithm [8].

We aim to find values for all parameters that minimize the error of the model, conditional on the number of voters and candidates. For this we define the error of our model as the (normalized) difference of the proportion of cyclic profiles, the proportion of simulation containing a Condorcet winner, and the proximity to single-peakedness using the voter based notion if possible and the candidate based notion. With these parameters, we argue that model captures the learning process. We then proceed to analyze convergence behavior under these optimal parameters, for this analysis, all configurations are run 100 times.

Given the best configurations, we will analyze the behavior of the model to understand the convergence on opinion. To this end, we measure the change in the trust matrix, as well as the distance between each voter's pre- and post-deliberation preferences using the KS and CS distances. We first aim to find the number of deliberative steps are needed for convergence, which we define as the moment where the largest change in the trust matrix is smaller than some ϵ. Then we hope to understand how individual voters' opinions change by looking at the final state of the trust matrix.

Finally, we use sensitivity analysis to investigate which parameters have the strongest effect on the model, using Sobol indices to get the first and second order effects. [5]

[5] This I have not had the time to implement in code yet, as it requires a little restructuring of how the model gets its parameters.

APPENDIX A

EXTENDED PROOFS

Finally, for CS, R_1 and R_j stay the same, while $R_1^{'} = c \succ a \succ b \succ \cdots \succ m$, resulting in $\text{Dist}_{\text{CS}}(R_1^{'}, R_j) = |2-2| + |1-3| + |3-1| = 4$.

BIBLIOGRAPHY

[1] James Fishkin, Valentin Bolotnyy, Joshua Lerner, Alice Siu, and Norman Bradburn. Can Deliberation Have Lasting Effects? *American Political Science Review*, 118 (4):2000–2020, November 2024. ISSN 0003-0554, 1537-5943. doi: 10.1017/S0003055423001363.

[2] Nesreen K. Ahmed, Jennifer Neville, and Ramana Kompella. Network Sampling: From Static to Streaming Graphs. *ACM Trans. Knowl. Discov. Data*, 8(2):7:1–7:56, June 2013. ISSN 1556-4681. doi: 10.1145/2601438.

[3] Maurice Queyranne. Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. *Operations Research Letters*, 4(5):231–234, February 1986. ISSN 0167-6377. doi: 10.1016/0167-6377(86)90007-6.

[4] Viswanath Nagarajan and Maxim Sviridenko. On the Maximum Quadratic Assignment Problem. *Mathematics of Operations Research*.

[5] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, March 2020. ISSN 1548-7105. doi: 10.1038/s41592-019-0686-2.

[6] Joshua T. Vogelstein, John M. Conroy, Vince Lyzinski, Louis J. Podrazik, Steven G. Kratzer, Eric T. Harley, Donniell E. Fishkind, R. Jacob Vogelstein, and Carey E.

Priebe. Fast Approximate Quadratic Programming for Graph Matching. *PLOS ONE*, 10(4):e0121002, April 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0121002.

[7] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[8] Gábor Erdélyi, Martin Lackner, and Andreas Pfandler. Computational Aspects of Nearly Single-Peaked Electorates. *Proceedings of the AAAI Conference on Artificial Intelligence*, 27(1):283–289, June 2013. ISSN 2374-3468. doi: 10.1609/aaai.v27i1.8608.

[9] PrefLib/preflibtools. PrefLib: A Library for Preferences, February 2025.

[10] Tomasz Przedmojski. *Algorithms and Experiments for (Nearly) Restricted Domains in Elections*. PhD thesis.