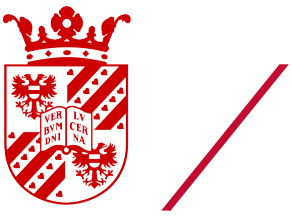
Faculty of Science and Engineering of the University of Groningen

On Partial Delegation in Liquid Democracy

Bachelor's Thesis in
Mathematics



university of
groningen

faculty of science
and engineering

Kane Harrison S3068218

Under the supervision of: prof. dr. D. Grossi
Additional assessor: dr. ir. B. Besselink
Date: January 2021

Abstract

Liquid democracy is a proxy voting system where proxies are delegable. We examine how partial delegation can be utilised to improve the quality of a social choice; we motivate three partial delegation mechanisms and examine how they converge under the DeGroot learning process. By simulating these delegation mechanisms, we demonstrate that splitting votes to all observed agents proportional to individual accuracy is the optimal delegation mechanism for smaller networks. In particular, this is true when comparing these delegation mechanisms to direct democracy and full delegation mechanisms.

Acknowledgements

It is important to acknowledge those individuals without whom completing my thesis would not have been possible.

Partway through the degree programme, circumstances required me to move home to the UK and complete studies from afar. Consequently, I need to offer my unwavering gratitude to Angelique and Alex who, through their friendship, offered a place to stay so I could sit exams. Without them, starting my thesis project would not have been possible, let alone completing it.

Furthermore, I need to offer my gratitude to Curtis and Scott who were willing to let me ramble enthusiastically about my chosen topic and provide points of feedback. Their friendship has inspired and motivated me to stay focused. Similarly, I need to thank my mother, as she always nurtured my love for mathematics.

I need thank Yuzhe who provided early advice about my topic. Importantly, I need to thank Professor D. Grossi and Dr. B. Besselink for agreeing to supervise the project. In particular, I am grateful to Professor Grossi for laying the groundwork for the topic and offering direction when it was needed.

Moreover, I am grateful to Mirjam, my academic advisor, for being a never-ending source of information, advice and support while I transitioned to studying from abroad and faced what seemed like endless consequences of Brexit.

Finally, I need to thank my life partner, Róisín, for her constant support and facilitating for me to have study time while we share caring responsibilities for our beautiful children.

Contents

Chapter 1

Introduction

Liquid democracy (Blum and Zuber 2015) brings compromise to two notions which, if considered to be of comparatively equal value, can lead to contradictions: direct democracy and representative democracy. A situation may occur where every agent in a social network votes directly on a decision and it contradicts a decision made by a group of representatives in the same network, leading to a problem which can be difficult to resolve. Liquid democracy draws on the strength of both notions by allowing each agent the option to vote directly or to nominate a proxy who will carry the weight of their vote. As an example, imagine a vote is being carried out among three individuals: Ava, Ben, and Charlie. The vote is about where they should go for dinner. They can each vote directly, so each vote has a weight of one, or they can nominate one of their friends as a proxy to vote on their behalf. Imagine that Ben is not a fussy eater and believes Ava is much more of an expert when it comes to food so he chooses her as his proxy. Ben can no longer vote directly, Charlie's vote still has a weight of one, but Ava's vote now has a weight of two.

The concept has been used for decision-making in the German Piratenpartei and the EU Horizons project WeGovNow (Boella et al. 2018) which has utilised the LiquidFeedback platform, as well as the Democracy Earth Foundation. Gradually, more papers are studying how aspects of liquid democracy pertain to social choice theory. Many of these investigate nuanced contexts for liquid democracy to study the resulting delegate structures and the impact on the quality of a social choice. These papers examine liquid democracy in relation to 'super-voters' (Gölz et al. 2018), recommender systems (Boldi

et al. 2009), rationality (Bloembergen, Grossi, and Lackner 2018), and more. However, there remains to be little literature on the impact of 'partial delegations'.

To illustrate how a partial delegation differs from the norm, consider the previous example with Ava, Ben, and Charlie choosing where to have dinner. Ben is still certain he does not know where would be best to eat and he believes Ava is a connoisseur when it comes to food, but he also believes that Charlie knows a little about food and that Charlie's knowledge differs from Ava's. In this scenario, Ben may consider lending 75% of his vote to Ava and 25% of his vote to Charlie so when the decision is made, Ben does not vote directly, Ava's vote has a weight of 1.75, and Charlie's vote has a weight of 1.25. In effect, when deciding how to distribute his vote of weight 1, Ben is using the cumulative rule and the alternatives over which he makes this decision can be considered to be all those within his personal social network, likely a subset of some larger network. Consequently, we would like to investigate the question: **In liquid democracy, what is the effect of partial delegation on the quality of a social decision?**

Important concepts will be defined to lay the groundwork for our model. We will motivate delegation mechanisms under a partial delegation framework with the hopes of comparing their effectiveness with each other, as well as to direct democracy and to a mechanism which maximises local utility, as described in Bloembergen et al (2018); this latter mechanism is to compare partial delegation against full delegation. Subsequently, we will consider the DeGroot learning process to investigate whether applying this to our delegation mechanisms will improve results. The motivation behind the DeGroot learning process is that the network observes the trust it has in its agents, and each agent adapts their opinion accordingly (DeGroot 1974). Analytic results will be outlined, including whether convergence of the DeGroot learning process can be determined and to what it converges.

Following the analytical results, simulations to test the performance of the delegation mechanisms and their DeGroot convergence matrices have been run. These simulations are tested over many graphs over four different typologies, two of which reflect real-world social connections. The results of the delegation mechanisms over these networks will be compared and discussed to draw conclusions about whether partial delegations are an effective

application of liquid democracy. We will be testing which delegation mechanisms maximise the probability of a 'correct' majority and which maximise the average utility for agents in a network.

Chapter 2

Preliminaries

2.1 Graph

Partial delegation within liquid democracy allows agents to delegate all or part of their vote to any other agents they can observe. These observed agents are considered to be in the agent's neighbourhood. An agent can retain (or, from a functional perspective, delegate to themselves) part or all of their vote. Subsequently, each agent is included in their own neighbourhood. Let $N \subset \mathbb{N}$ be the number of agents in a network. Let $R \subset \mathbb{N}^2$ be a binary relation over the set of agents where $(i, j) \in R$ indicates that j belongs to i's neighbourhood, which can be denoted as $j \in R(i)$. Note that for every $i \in N$, we have $(i, i) \in R$ which means R is a reflexive relation. Furthermore, $(i, j) \in R$ if and only if $(j, i) \in R$ which means R is a symmetric relation.

The network of agents can be represented as an undirected graph $G = \langle N, R \rangle$. The information contained in a graph can be conveyed pictorially as a collection of nodes representing the agents and edges connecting agents who belong in each other's neighbourhoods, or it can be contained in an adjacency matrix where $G_{ij} = 1$ if $j \in R(i)$ and $G_{ij} = 0$ otherwise.

2.2 Type

Liquid democracy involves a network of agents making a choice or sequence of choices. It will be assumed that there is a 'ground truth' choice which is

objectively preferable for each agent for any choice. This ground truth will be referred to as the agent's type, τ_i. Only binary choices will be considered, so it can be assumed that $\tau_i \in \{0, 1\}$.

Agents are unaware of their type but will have an associated accuracy, q_i, which estimates the likelihood an agent will correctly vote for their type. It will be assumed that agents have at least as good as random chance of correctly guessing their type so $q_i \in [0.5, 1]$. This paper will assume homogeneity, therefore, all agents will have the same type. This is counter to models explored in other papers which include deterministic type profiles, where agents may have different types but the types are certain and can be recorded in a vector, and probabilistic type profiles, where p_{ij} represents the probability that agent i and agent j are of the same type according to a given probabilistic distribution.

2.3 Delegation

Given a graph, it is important to determine a delegation profile, D, to demonstrate how agents distribute the weight of their vote among their neighbours. This will be represented as a stochastic matrix. That is, for agent i, row i will represent the agent's spread of their vote; consequently, every row will sum to 1.

2.3.1 Mechanisms

Any delegation profile is permissible so long as $D_{ij} \in [0, 1]$ and D is a stochastic matrix. As such, it is important to motive mechanisms to decide how each agent assigns their delegations. It is assumed that agents have knowledge of each other's accuracy so the most appropriate delegation mechanisms will take as input the graph, G, and the accuracy vector, q, to return a delegation profile. An example of a trivial delegation profile is that of direct democracy, each agent retains their full weighted-1 vote leading to $D = I_N$.

2.3.2 DeGroot Learning

DeGroot learning is a social learning process where a network of agents observe the trust they each hold in one another to converge to an opinion. In its original context, the process uses an initial opinion represented as a probability vector with values between 0 and 1, which can be denoted as $p(0)$, and a stochastic trust matrix T where T_{ij} represents the trust that agent i has in the opinion of agent j. The learning process is motivated by the idea that agent i will adjust their opinion in accordance with their level of trust of other agents and those agents' opinions. Let t represent a time-step,then this process can be repeated to derive the formula

$$p(t) = T^t p(0).$$

2.4 Network Types

Networks can have different underlying structures which may impact the network's behaviour. This paper will consider four typical network structures:

- the random network; each pair of agents has a specified probability of belonging to each other's neighbourhoods. (Erdös and Rényi 1959)
- the regular network; each agent will have the same degree, that is, they will have the same number of agents belonging to their neighbourhood.
- the small-world network; average path length connecting agents will be small and there will be high clustering. (Watts and Strogatz 1998)
- the scale-free network; there is a power law degree distribution. (Barabasi and Albert 1999)

For comparison of results, this paper will also consider totally connected networks in which every agent belongs to every agent's neighbourhood.

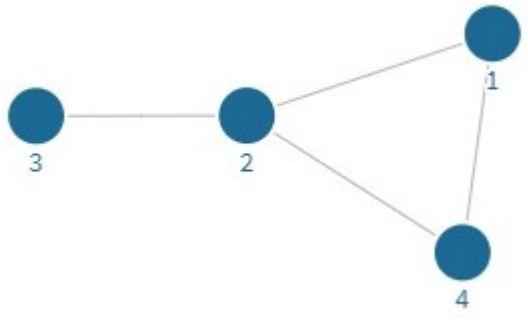
Chapter 3

Model

3.1 Graph and Adjacency Matrix

Graphs of the different typologies are generated by different Matlab scripts. The random network and small-world network will be generated by WattsStrogatz.m; the former is achieved by setting the rewiring probability to 1 and the latter is achieved by setting it to any value between 0 and 1. For the sake of this paper, the rewiring probability will be set to 0.5. The regular network is generated by createRegGraph.m (Pundak 2020). The scale-free network is generated by SFNW.m (George 2020). Every graph can be represented by its adjacency matrix.

Figure 3.1: Example Graph



Consider the example graph presented in figure 3.1; this graph can be represented with an adjacency matrix as follows:

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Special note should be made that all graphs discussed in the context of liquid democracy will be reflexive as all agents are able to retain the weight of their vote. As such, edges connecting nodes to themselves in the graphs will be omitted for visual clarity but will be represented by a 1 in the respective adjacency matrix.

3.2 Accuracy

Each agent needs a randomised accuracy associated. These accuracies are randomised according to a normal distribution, $\mathcal{N}(0.75, 0.1)$, by the Matlab function accuracy.m (Appendix A). Conventionally, the normal distribution will have no maximum or minimum values. However, there are examples where maximum and minimum values exist by construction which fit a normal distribution, such as exam scores. The function seeks to emulate this example; where the random number generated falls outside of the $[0.5, 1]$ range, the function will either subtract it from 1 (if it is below 0.5) or subtract it from 2 (if it is above 1). This ensures that all accuracies fall within the desired region while fitting a normal distribution.

3.3 Delegation Mechanisms

Given a graph which tells us each agent's neighbourhood and an accuracy associated with every agent, it becomes important to motivate how each agent will spread their weighted 1 vote among themselves and their neighbourhood. This model posits three methods utilising partial delegations:

- **Delegation Mechanism 1**: Each agent observes their own accuracy and the accuracy of their neighbours. Provided with this information,

the agent spreads the weight of their vote proportionally according to accuracy among all agents they can observe. (Appendix F)

- **Delegation Mechanism 2**: Provided with the knowledge of accuracies of all agents in their neighbourhood, an agent spreads the weight of their vote among those they can observe whose accuracy is equal to or greater than their own (including themselves). This emulates a delegation mechanism discussed by Kahng et al (2018) which randomises approved voters, but this takes a partial delegation approach to the concept.(Appendix G)

- **Delegation Mechanism 3**: A stronger version of delegation mechanism 2, an agent spreads the weight of their vote among those they can observe whose accuracy is *greater* than their own if they exist. Under this mechanism, the only agents who delegate to themselves are ones who have no agents with a greater accuracy in their neighbourhood. (Appendix H)

The delegation profiles generated by these mechanisms are weighted directed graphs. In particular, delegation mechanism 3 results in a profile where delegated votes will flow to agents with the highest accuracy in their neighbourhood, which will become sink nodes.

To illustrate these delegation mechanisms with an example, consider figure 3.1 with accuracies

$$q = \begin{bmatrix} 0.6 \\ 0.5 \\ 0.7 \\ 0.9 \end{bmatrix}.$$

The mechanisms will lead to the respective delegation profiles:

$$D_1 = \begin{pmatrix} \frac{3}{10} & \frac{1}{4} & 0 & \frac{9}{20} \\ \frac{2}{9} & \frac{5}{27} & \frac{7}{27} & \frac{1}{3} \\ 0 & \frac{5}{12} & \frac{7}{12} & 0 \\ \frac{3}{10} & \frac{1}{4} & 0 & \frac{9}{20} \end{pmatrix}$$

$$D_2 = \begin{pmatrix} \frac{2}{5} & 0 & 0 & \frac{3}{5} \\ \frac{2}{9} & \frac{5}{27} & \frac{7}{27} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$D_3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ \frac{3}{11} & 0 & \frac{7}{22} & \frac{9}{22} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Two delegation mechanisms which assign full delegates rather than partial delegates will be used for comparing results. The first represents direct democracy; each agent retains their full weighted vote of 1 and inherits their own accuracy (Appendix I). The second is adapted from Bloembergen et al (2018) and has each agent assign their full weighted vote to the agent in their neighbourhood who has the highest accuracy with the motivation to locally maximise their utility (Appendix J).

3.4 Delegation Paths, Gurus, and Utility

Once a delegation profile is obtained, in order to calculate either the expected utility of the agents or the probability of a correct majority, it is important to calculate votes weighted by all delegations that each agent communicates to the voting mechanism. Consequently, a guru profile needs to be determined. In liquid democracy without partial delegations, each agent will have a guru who communicates the full weight of their vote to the voting mechanism: this can be the agent themselves, it can be an agent in their neighbourhood or, if the agent to whom they chose to delegate delegates further, the agent who delegates to themselves along this path. In this case, each agent will have exactly one guru and one agent can behave as the guru for many other agents. The weight of the vote that a guru communicates to the voting mechanism

is the sum of the weighted-1 votes of the agents whom the guru represents.

When considering partial delegation, each agent can have more than one guru and each has a partial weight of the agent's weighted-1 vote. That is, the sum of the votes that the gurus communicate to the voting mechanism on the agent's behalf will sum to 1 (if there are no cycles). Similarly, the weight of the vote that a guru communicates to the voting mechanism will be the sum of the partial weights of the votes of each agent the guru represents.

Given the transitive nature of partial delegations, an agent may delegate some of their vote to one agent who then delegates some of their vote (including the weight acquired from the previous agent) to another agent in the network, and this process may continue. It becomes important to calculate how much of any agent's vote is delegated to any other agent via any path of any length. An important observation is that the maximum path length between a delegating agent and their guru which does not permit a cycle is N. Consider $D^{\mathrm{p}-k}$ to be the path profile where $D_{ij}^{\mathrm{p}-k}$ denotes the sum of the partial weights of agent i delegated to agent j by all paths of length k, where $i, j, k \in \{1, \dots N\}$. The weight of a vote retained by an agent can be considered a path-0 delegation, we denote $D^{\mathrm{p}-0}$ to be a zero matrix with diagonal elements equal to the diagonal elements of D. The path profile $D^{\mathrm{p}-1}$ corresponds to all partial delegations of path 1 and is equal to the delegation profile, D, with its diagonal elements set to 0. These diagonal elements correspond to the amount of their vote which each agent retains and as such, they are communicated directly to the voting mechanism rather than fed into any further paths. Consequently, these diagonal elements represents a cycle and must be removed before feeding into any other paths. The path profile $D^{\mathrm{p}-k}$ is calculated by multiplying $D^{\mathrm{p}-1}$ by itself k times and removing the diagonal elements at each iteration so cycles are removed from the process. The partial weights which are removed due to cycles never land on a guru and thus are never communicated to the voting mechanism. Let D^{p} be a path profile where D_{ij}^{p} denotes the sum of the partial weights of agent i delegated to agent j by all paths of any length. This can be calculated by taking the sum

$$D^{\mathrm{p}} = \sum_{k=0}^{N} D^{\mathrm{p}-k}.$$

The guru matrix, D^{g}, is calculated by multiplying the jth column of D^{p} by D_{jj}, the weight of their own vote that the guru retains. Each element in the guru matrix, D^{g}_{ij} corresponds to the total weight of agent i's vote that agent j communicates to the voting mechanism delegated by any path. If there are no cycles, D^{g} will be a stochastic matrix.

Utility is a focus in the Bloembergen et al (2018) paper's approach to liquid democracy. Whereas it is not a prime area of focus in this paper, it is fruitful to compare the utility generated by the different delegation mechanisms and what they seem to suggest from a practical viewpoint.

This paper is assuming homogeneity and in liquid democracy without partial delegations, the utility of an agent will be equal to the accuracy of their guru. This is the probability that the weight of the agent's vote is communicated to the voting mechanism correctly identifying the agent's type. In the context of liquid democracy with partial delegations, a utility vector can be found by the formula

$$u = D^{\mathrm{g}} q.$$

Note that u_i is the sum of the weights of their vote delegated to each guru multiplied by the respective guru's probability of correctly identifying the ground truth for the network.

To illustrate, consider figure 3.1 equipped with the following delegation profile and accuracies

$$D = \begin{pmatrix} 0.1 & 0.4 & 0 & 0.5 \\ 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{q} = \begin{bmatrix} 0.9 \\ 0.6 \\ 0.7 \\ 0.9 \end{bmatrix}.$$

These leads to the following step matrices:

$$D^{\mathrm{p}} = \begin{pmatrix} 0.1 & 0.4 & 0.12 & 0.78 \\ 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$D^{\mathrm{g}} = \begin{pmatrix} 0.1 & 0 & 0.12 & 0.78 \\ 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Finally, utility is found by multiplying D^{g} by the accuracy vector q. For the example, we find

$$\mathbf{u} = \begin{bmatrix} 0.876 \\ 0.84 \\ 0.7 \\ 0.9 \end{bmatrix}.$$

3.5 DeGroot Learning

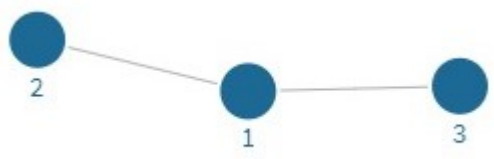
The DeGroot learning process has analogues in the liquid democracy context so it bears investigating as to whether this may improve the results of any delegation mechanism. A delegation profile behaves as an initial trust matrix among a network of agents and the opinion which they will attempt to converge on is their trust in each other. That is, using notation set out in the preliminaries, we have that $T = D$ and that $p = D_j$ where D_j denotes the jth column of D and denotes the network's trust in j's opinion. Note that an agent i's trust in agent j is a partial delegation between 0 and 1, and therefore, returns a similar formula to the standard DeGroot learning process. However, there are two important differences between these two contexts. The first is that because both the trust matrix and the opinion vector are analogues of the delegation profile, the opinion vector is the trust matrix. Consequently, the trust matrix will evolve at each iteration leading to $D^{t+1} = D^{2^t}$.

The second important difference is that the process needs to be adapted as it is not required for a network to be totally connected; if agent i does

not observe agent j in their neighbourhood, then it is impossible for D_{ij} to become nonzero at any stage. However, applying the DeGroot learning process as standard could lead to this occurring. To resolve this, at every iteration any nonzero D_{ij} where $j \notin R(i)$ needs to be set to zero then the ith row needs to be normalised so that the agent still delegates the full weight of their vote. Unless a row becomes 0 at any point, every iteration during the DeGroot learning process should continue to be stochastic.

In adapting the DeGroot learning process for liquid democracy, there is one final question to be addressed. That is whether an agent has an awareness of the network structure. To illustrate, consider the network in figure 3.2.

Figure 3.2



Let the agents have the associated accuracy vector

$$q = \begin{bmatrix} 0.5 \\ 0.6 \\ 0.9 \end{bmatrix}.$$

Under delegation mechanism 1, this would lead to a delegation profile

$$D = \begin{pmatrix} \frac{5}{20} & \frac{6}{20} & \frac{9}{20} \\ \frac{5}{11} & \frac{6}{11} & 0 \\ \frac{5}{14} & 0 & \frac{9}{14} \end{pmatrix}.$$

If agent 1 does not acknowledge that agent 2 and agent 3 do not belong to each other's neighbourhoods when adapting their delegations in accordance to their opinions, then there is no change to how each iteration of the DeGroot learning process is calculated, and the process will converge to a state where agent 1 awards themselves a higher partial delegation than they

award agent 3 despite agent 3 having greater accuracy and being awarded the higher delegation in the initial delegation profile. However, if we wished to model the DeGroot learning process so that agent 1 shows an awareness of the network structure when adjusting their trust in the agents, there is an additional step to iterating the process.

Assume we wish to determine how agent i uses their knowledge of the network to adjust their trust in agent j. The first step would be to determine the largest subnetwork where both agent i and agent j are totally connected; that is, all agents in the subnetwork belong to their neighbourhoods. This step may lead to rows which sum to less than 1 including the rows for agents i and j. Consequently, all rows should be normalised so their sum is one; each row representing an agent k for all $k \in R(i), R(j)$ is divided by the sum of the row resulting in a stochastic matrix. It may be the case that not all agents in this subnetwork are totally connected. Consequently, each row representing an agent k needs to be multiplied by the sum of the (potentially normalised) trust agent i has in the agents observed by both agent i and agent k - intuitively, this step weights agent k by the trust agent i has in their shared subnetwork. Note that the matrix would no longer be stochastic, though by definition of how the subnetwork is defined, neither rows representing agent i or j would be weighted. At this stage, the resulting matrix should be squared for agent i to adjust their opinion. The rows representing the trust values for agent i will sum to 1 and demonstrate how the agent's trust will evolve in accordance to the DeGroot learning process for this particular subnetwork.

The intuition behind this process is that by taking the subnetwork with all agents that agent i can observe who also observe agent j, that agent i only considers the relevant opinions when adjusting their trust. With a common frame of reference, it is possible to use their involve to decide how their trust should evolve in the wider context of the network. As agent i will appear in every subnetwork, this is the best choice for a frame of reference, so repeating this process for all agents k that i can observe in a network, the ratio between the trust that agent i has in itself and in agent k should be recorded along with the restraint that the sum of all of agent i's trust values will equal 1 to create a system of equations. This system will include N unknowns with N equations so will be uniquely solvable.

Chapter 4

Analytic Results

4.1 Mixed Extension-Partial Delegation Equivalence

Provided with a graph, an accuracy vector, and a delegation profile, there are different contexts in which results can be discussed. Importantly, the delegation profile can be considered to be either the partial delegation of agent i's vote to agent j for a one-event vote or it can be considered as the probability that agent i will delegate the full weight of their vote to agent j in any vote which may occur. The context has important distinctions when discussing cycles and utility.

When determining the path matrix, the delegation profile is multiplied by itself and the diagonal elements are removed at each stage. In a one-vote event, these diagonal elements correspond to the weight of each agent's vote which is lost through a partial cycle and does not get communicated to the voting mechanism. When the delegation profile is considered as a probabilistic distribution of delegations, these diagonal elements represent the probability that a given agent becomes trapped in a cycle, and the entire weight of their vote is not communicated to the voting mechanism.

Similarly, the utility vector in a one-event vote is equivalent to an expected utility vector when the delegation profile is taken as a probabilistic distribution. However, in both cases, the probability of a correct majority, which we will call the accuracy of a majority, requires the same calculation and bears the same meaning.

4.2 Trivial Examples

When discussing what may be the optimal strategy for a network of agents to maximise their probability of choosing the ground truth in a vote, there are two trivial cases which should be acknowledged. The first is when every agent except one has an accuracy of 0.5 regardless of the number of agents in the network. In this example, the optimal strategy for choosing the ground truth in a vote will always be for each agent with 0.5 accuracy to delegate the full weight of their vote on a path which leads to the agent with higher accuracy, provided that a path exists.

The second trivial result is where at least one agent has an accuracy of 1; this has a practical real-world example which is a group of non-experts with one expert who has field-specific knowledge. In such a case, it is trivial to see that the probability of choosing the ground truth is maximised by all agents delegating on a path which leads to the agent of accuracy 1, provided that such a path exists. In this example, choosing the ground truth would be guaranteed.

Both of these examples involve delegating to an agent of higher accuracy; as later results demonstrate, this will not always be the optimal strategy. These trivial examples are avoided in simulations as we consider accuracy to take values between 0.5 and 1 through a normal distribution - whereas it is possible for an agent's accuracy to assume these values, the chances of it occurring are negligible.

4.3 Accuracy of a Majority

This paper is primarily concerned with which delegation mechanism may yield the greatest probability of a majority of agents voting for the ground truth. The way this is calculated is by considering only the set of gurus and their associated weights by summing each agents respective column in the guru profile, D^g. Subsequently, the threshold for a majority is taken by summing these weights and dividing by two. If there are no cycles, this will be equal to $\frac{N}{2}$; however, cycles are permitted in this model. Finally, the probability of a majority selecting the ground truth is found by calculating the probability of each possible guru vote combination using the accuracy vector, then taking the sum of the probabilities of the combinations where the weight of the gurus choosing the ground truth exceeds the threshold. This is the concept which motivates the MajProb.m function (Appendix N).

4.4 DeGroot applied to Delegation Mechanisms

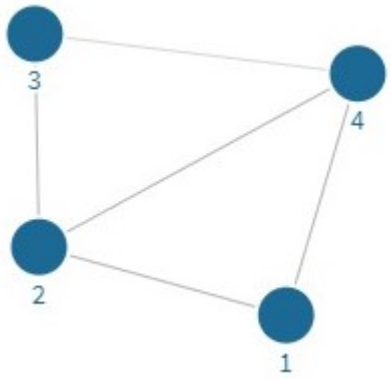
With the standard DeGroot learning process, it is not guaranteed that the process will converge. It is required that every subset of nodes which are strongly connected are closed and aperiodic (DeGroot 1974). In the context of liquid democracy, each strongly connected subset of agents is closed by definition - it is not possible for an agent to delegate to an agent who is not in their neighbourhood. Therefore, in this context, it is only required for the process to be aperiodic.

With the defined delegation mechanisms, their behaviour and convergence can be ascertained analytically. For all three cases, convergence is guaranteed and the delegation profile to which the process converges can be determined by investigating the convergence behaviour. Two approaches for the DeGroot learning process as applied to liquid democracy have been discussed. The chosen approach does not affect convergence for any of the delegation mechanisms outlined and whereas it does not affect behaviour for the delegation mechanism 3, the choice of approach does affect the behaviour for delegation mechanisms 1 and 2.

4.4.1 Convergence of Delegation Mechanism 1

The approach for the DeGroot learning process chosen does not affect whether delegation mechanism 1 converges as it will always converge under both approaches. However, the choice of approaches does affect the behaviour of convergence. To illustrate the behaviour of convergence under both approach consider the network in figure 4.1.

Figure 4.1



Let the agents be associated with the accuracy vector

$$q = \begin{bmatrix} 0.5 \\ 0.7 \\ 0.6 \\ 0.7 \end{bmatrix}.$$

Under delegation mechanism 1, this will lead to the delegation profile

$$D = \begin{pmatrix} \frac{5}{19} & \frac{7}{19} & 0 & \frac{7}{19} \\ \frac{1}{5} & \frac{7}{25} & \frac{6}{25} & \frac{7}{25} \\ 0 & \frac{7}{20} & \frac{3}{10} & \frac{7}{20} \\ \frac{1}{5} & \frac{7}{25} & \frac{6}{25} & \frac{7}{25} \end{pmatrix}.$$

Let D_t^* be the tth iteration with $D_0^* = D$ and let D^* denote the convergence matrix of the DeGroot learning process as t approaches infinity. Let $Q_i = \sum_{j \in R(i)} q_j$.

Conjecture 4.4.1. *Assume agents have no knowledge of the network structure. Delegation mechanism 1 under the DeGroot learning process will converge to*

$$D_{i,j}^* = \frac{\frac{\sum_{m \in R(i), R(j)} q_m}{Q_i} \times \sum_{k \in R(i)} \frac{q_j}{Q_k}}{\sum_{m \in R(i), R(j)} \left(\frac{\sum_{n \in R(i), R(m)} q_n}{Q_i} \times \sum_{k \in R(i)} \frac{q_m}{Q_k} \right)}.$$

To elaborate how to interpret this formula, D_{ij}^* is equal to the sum of column j for agent i's subnetwork multiplied by the sum of row j from agent i and agent j's shared subnetwork divided by the sum of the sum of all columns from agent i's subnetwork multiplied by their respective rows from their shared network with agent i. This result was ascertained through numerical investigation; several networks and initial delegation profiles were iterated through the DeGroot learning process to observe the convergence behaviour. For the provided example, we calculate

$$D^* = \begin{pmatrix} \frac{5}{19} & \frac{7}{19} & 0 & \frac{7}{19} \\ \frac{2394}{17503} & \frac{12145}{35006} & \frac{2964}{17503} & \frac{12145}{35006} \\ 0 & \frac{7}{20} & \frac{3}{10} & \frac{7}{20} \\ \frac{2394}{17503} & \frac{12145}{35006} & \frac{2964}{17503} & \frac{12145}{35006} \end{pmatrix}.$$

Theorem 4.4.1. *Assume agents have full knowledge of their subnetwork structure. Delegation mechanism 1 under the DeGroot learning process will be invariant.*

Proof. Observe that delegation mechanism 1 results in a delegation profile with

$$D_{i,j} = \frac{q_j}{Q_i}.$$

Consider the largest subnetwork where agent i and agent j are totally connected, $D^{i,j}$. Normalising this network would lead to, for all $k, m \in R(i), R(j)$,

$$D_{m,k}^{i,j} = \frac{q_k}{Q_m} \times \frac{Q_m}{Q_{i,j,k}}$$

where $Q_{i,j,k} = \sum_{n \in R(i),R(j),R(k)} q_n$. Weighting the rows for all agents k not totally connected will lead to, for all $k, m \in R(i), R(J)$,

$$D^{i,j}_{m,k} = \frac{Q_{i,j,k}}{Q_{i,j}} \times \frac{q_k}{Q_m} \times \frac{Q_m}{Q_{i,j,k}} = \frac{q_k}{Q_{i,j}}.$$

Next, consider the square of this matrix, as we only hope to find the adjusted trust of agent i in agent j, this leads to

$$\sum_{k \in R(i),R(j)} \frac{q_j q_k}{Q_{i,j} Q_{i,j}} = \frac{q_j Q_{i,j}}{Q_{i,j} Q_{i,j}} = \frac{q_j}{Q_{i,j}}$$

where the first step uses that $\sum_{k \in R(i),R(j)} q_k = Q_{i,j}$. Consequently, the ratio of trust agent i has in agent j with respect to themselves is

$$D_{i,j} = \frac{q_j}{q_i} D_{i,i}.$$

The linear system of equations to evolve agent i's trust becomes $D_{i,j} = \frac{q_j}{q_i} D_{i,i}$ for every $j \in R(i)$ and $\sum_{j \in R(i)} D_{i,j} = 1$. If we set $D_{i,i} = q_i$, then $D_{i,j} = q_j$ for every $j \in R(i)$. Finally, the restriction $\sum_{j \in R(i)} D_{i,j} = 1$ requires that agent i's trust values be divided by $\sum_{j \in R(i)} q_j = Q_i$. Therefore, the solution for linear the system of equations is $D_{i,j} = \frac{q_j}{Q_i}$.

Therefore, the trust that agent i has in any agent j is unchanged. As agent i and agent j are chosen arbitrarily, delegation mechanism 1 is invariant under the DeGroot learning process resulting in $D^* = D$.

☺

4.4.2 Convergence of Delegation Mechanism 2

The convergence of delegation mechanism 2 depends on which approach to the DeGroot learning process is considered.

Conjecture 4.4.2. *Assume agents have no knowledge of the network structure. Delegation mechanism 2 under the DeGroot learning process will converge to*

$$D^*_{i,j} = \begin{cases} 1, & \textit{if } j \textit{ is not a sink and has highest trust in themselves among} \\ & \textit{agent } i\textit{'s neighbourhood} \\ 0, & \textit{for all other } j\textit{s which are not sinks} \end{cases}$$

If agent i has at least one sink in their neighbourhood, their vote will converge to being split among these sinks proportional to their accuracy.

This result was obtained by observing the behaviour of the process numerically. All agents who are not sinks will see their self-trust converge to zero as the process is iterated. Intuitively, this can be explained by noting under this delegation mechanism, the agents only receive reciprocated trust from themselves, so unless they are a sink, this will tend to zero. This results in a convergent matrix where every agent delegates their full-weighted vote along a path which ends with at least one sink.

There are three specific behaviours which are important to note when discussing this result. First, assume we have a five agent network. Agent 1 can observe agents 2 and 3 who cannot observe each other and neither of which is a sink. Assume that $q_3 > q_2 > q_1$ and due to their relationship to other agents, agent 2 starts with a higher self-trust compared to agent 3. In this scenario, despite agent 3 having the higher accuracy, agent 1 will converge to awarding their full-weighted vote to agent 2. Iterating through the process, we observe that agent 1's trust in agent 3 may increase initially, but the growth at which it increases will decelerate until eventually it decreases to zero - this can be attributed to the speed at which an initial small difference between the self-trust of agents 2 and 3 widens as the process is repeated.

A second important observation uses a similar structure except agent 2 observes two unconnected agents with a higher accuracy and agent 3 observes three unconnected agents with a higher accuracy. If accuracies are defined such that agent 2 and agent 3 begin with the same self-trust, their self-trust will decrease at the same rate despite agent 3 having more agents in their neighbourhood with a higher accuracy to whom they can delegate part of their vote. Finally, the behaviour which occurs for agents with multiple sinks was obtained through numerical experiments.

Conjecture 4.4.3. *Assume agents have full knowledge of their subnetwork structure. Delegation mechanism 2 under the DeGroot learning process will*

converge to

$$D^*_{i,j} = \begin{cases} 1, & \textit{if agent } j \textit{ has the highest accuracy in agent } i\textit{'s neighbourhood} \\ 0, & \textit{otherwise} \end{cases}$$

Intuitively, by design of the delegation mechanism, if all agents adjust their opinions with knowledge of the network structure, their opinion will converge to awarding their neighbour with the highest accuracy. Consider figure 3.2 with the following accuracy vector:

$$q = \begin{bmatrix} 0.6 \\ 0.8 \\ 0.9 \end{bmatrix}.$$

This leads to the following delegation profile:

$$D = \begin{pmatrix} \frac{6}{23} & \frac{8}{23} & \frac{9}{23} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

One iteration of the DeGroot learning process where agents have knowledge of the network structure yields

$$D^*_1 = \begin{pmatrix} \frac{36}{457} & \frac{196}{457} & \frac{225}{457} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

This first iteration demonstrates that agent 1's trust in itself and agent 2 is trending downwards, while their trust in agent 3, the most accurate agent, is trending upwards. Repeated iterations further demonstrate this trend. The convergence of this delegation mechanism under this approach is identical to the full delegation mechanism which maximises local utility in Bloembergen et al (2018), which is being used in this paper for a base comparison.

4.4.3 Convergence of Delegation Mechanism 3

Contrary to the previous delegation mechanisms, the second approach to the DeGroot learning process cannot be applied to delegation mechanism 3. This is because the mechanism uses an agent's trust in themselves as a frame of reference to adjust all other trust values; however, this delegation mechanism presumes an agent has zero trust in themselves. Though the second approach to the DeGroot learning process cannot be applied, we can still observe the convergence behaviour under the first approach.

Theorem 4.4.2. *Assume agents have no knowledge of the network structure. Delegation mechanism 3 under the DeGroot learning process will converge* $D^*_{i,j} = 0$ *if an agent* i *has no sinks in their neighbours. If agent* i *has at least one sink in their neighbourhood, their vote will converge to being split among these sinks proportional to their accuracy.*

Proof. Assume an agent i has no sinks in their neighbourhood. By design, no agent in agent i's neighbourhood has self-trust. Consequently, one iteration of the DeGroot learning process leads to this agent awarding zero weight of their vote, which additionally means it is not possible for their trust to be normalised. Next, assume that agent j has N sinks and M agents with higher accuracy in their neighbourhood. Clearly we have $M > N$; number the agents arbitrarily so sinks are numbers from 1 through to N and all other agents are numbered $N + 1$ through to M. For a given sink n, where $n \in 1, \ldots, N$, their trust from agent i under delegation mechanism 3 will be

$$D_{i,j} = \frac{q_n}{Q_M},$$

where Q_M is the sum of the accuracies for all agents in agent i's neighbourhood with higher accuracy. In the first iteration, agent i's trust in all sinks will be multiplied by one and all other trust values will becomes 0. The next step to obtain the second iteration is to normalise agent i's trust values so they sum to 1. Consequently, for a given sink n, we calculate

$$\frac{\frac{q_n}{Q_M}}{\sum_{k=1}^{N} \frac{q_k}{Q_M}} = \frac{q_n}{Q_M} \times \frac{Q_M}{Q_N} = \frac{q_n}{Q_N}$$

where Q_N is the sum of accuracies for sinks.

☺

Chapter 5

Simulations

5.1 Set Up

Simulations were run to compare how the delegation mechanisms and their DeGroot counterparts affect the probability that a random network of agents with random accuracy belonging to a normal distribution, $\mathcal{N}(0.75, 0.1)$, has a majority in favour of the ground truth. Additionally, this is compared to the performance of direct democracy and a mechanism motivated by maximising local utility.

The code which calculates the accuracy of a majority for a given delegation profile, MajProb.m (Appendix N), needs to calculate every possible vote combination among a set of gurus. As such, the time it takes for the function to run scales proportional to 2^{N*} where $N*$ is the number of gurus in a delegation profile. The DeGroot counterpart of delegation mechanisms 1 and 2 and the maximising local utility mechanism on average have fewer gurus and can be computed for larger networks; however, other mechanisms involve all agents acting as a guru, so the calculation is too time-expensive and this limits the size of networks which can be considered for simulations.

The networks considered were $N = 4$ with two cases: average degree 2 and totally connected. Additionally, networks with $N = 16$ are analysed with average degrees 4, 8, 12, and the totally connected case. For average degrees 4 and 8, simulations were run for all network typologies discussed: random, regular, small-world, and scale-free. For average degree 12, only the first

three were analysed as a scale-free network is not possible for that average degree with a network of only 16 agents. For each example, 250 simulations were run (10 randomised graphs with 25 randomised initialisations on each).

For each case, four charts have been generated:

- A box plot to show the range of values taken by the accuracy of the majority over all simulations for each delegation mechanism.
- A box plot to show the range of average network utilities over all simulations for each delegation mechanism.
- A bar chart to show the mean accuracy of a majority for each delegation mechanism in the simulations.
- A bar chart which shows the percentage of the simulations for which each delegation mechanism is the (joint-)optimal strategy.

All of these charts are provided in the Appendix and those which are most relevant are provided to discuss results.

5.2 Results

For every set up for which a simulation was run, delegation mechanism 1 appears to be the optimal strategy for the network.The typology of the network has little impact on the results and did not seem to change the order of the results for any network size or any average degree. The charts for the small-world networks will be used here to discuss the results but the pattern is the same among all typologies.

Consider figures 5.1, 5.2, 5.3 and 5.4. In both cases, delegation mechanism outperforms its DeGroot counterpart, which in turn outperforms direct democracy, which outperforms all other delegation mechanisms. Delegation mechanism 1 consistently outperforms all other mechanisms as can be seen in figures 5.5 and 5.6.

Figure 5.1

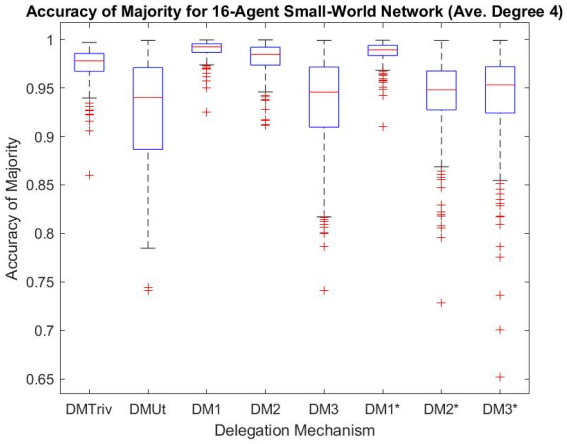


Figure 5.2

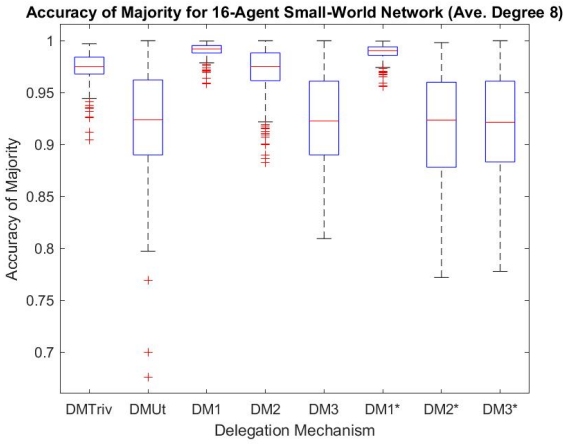


Figure 5.3

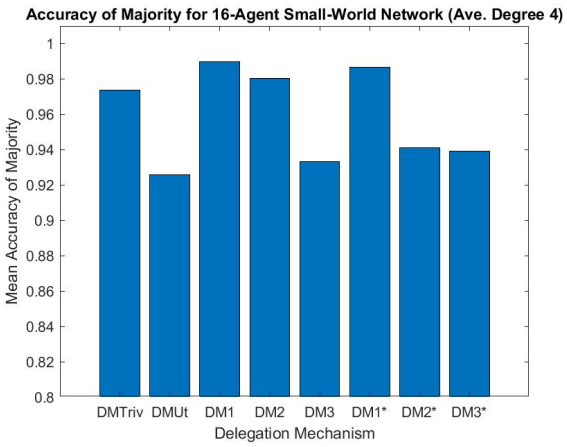


Figure 5.4

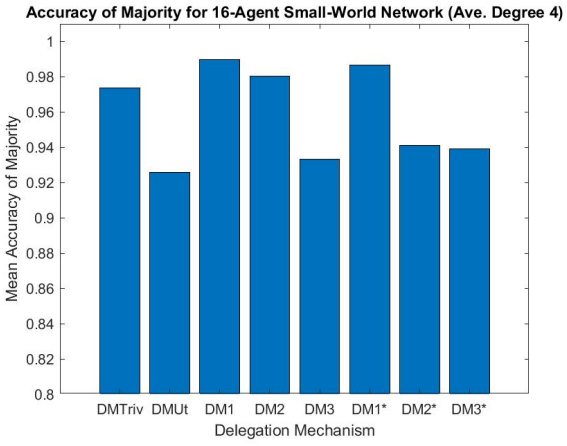


Figure 5.5

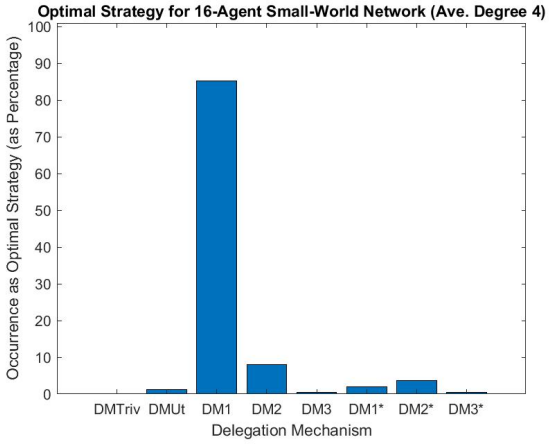
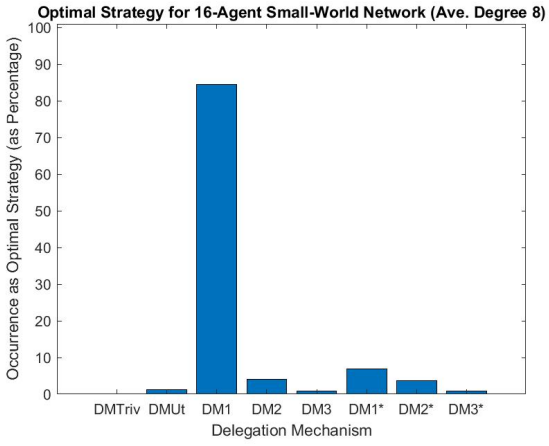


Figure 5.6



Comparing the results for $N = 16$ to $N = 4$, consider figures 5.7 and 5.8. These results seem to suggest that delegation mechanism 1 and its DeGroot counterpart, as well as direct democracy, seem to improve the accuracy of a majority as N increases whereas all other delegation mechanisms seem to worsen as N increases. This behaviour for direct democracy is a well-established result and can be ascertained from Condorcet's jury theorem (Condorcet 1785).

Use figures 5.9, 5.10, 5.11 and 5.12 in conjunction with figures 5.1, 5.2, 5.3 and 5.4 above to compare behaviour as average degree increases. That is, as a network becomes more 'socially connected'. Direct democracy is unaffected as the connectivity of the network does not affect how agents vote. However, delegation mechanism 1 and its DeGroot counterpart appear to improve with greater social connectedness whereas all other delegation mechanisms worsen.

With regards to utility, maximising utility locally does not necessarily maximise utility overall for an agent, as there may be an agent with greater accuracy than their guru had they delegated along a path with an agent in their neighbourhood with lower accuracy. However, it is the best performing in maximising the average utility for a network. The worst performing for utility is delegation mechanism 1 which decreases as N increases and decreases as average degree increases. In other words, as delegation mechanism 1 improves the accuracy of a majority, it worsens the network's average utility. This is due to the motivation behind the mechanism, as every agent splits their vote among all their neighbours and since they receive a partial weight from every neighbour, every agent has some weight of their vote not communicated to the voting mechanism because it is lost in a partial cycle. The more connected an agent, the more of the weight of its vote is lost in partial cycles, the lower their utility. It is interesting this occurs in the case that the accuracy of a majority is maximised.

The final important comment on the results of the simulations is that when considering all charts is that the size of the box for delegation mechanism 1 and its percentage in the cases where it is the optimal strategy for a given graph and accuracy vector show that it is the delegation mechanism to deliver most consistently.

Figure 5.7

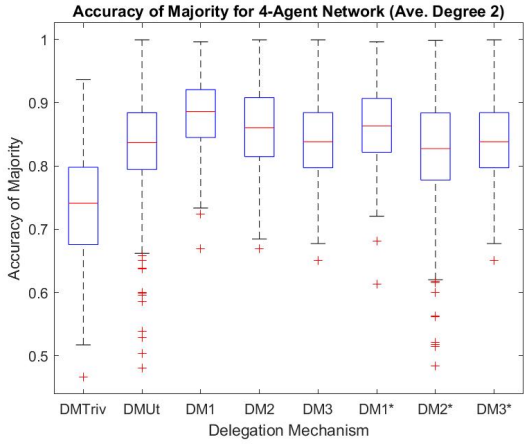


Figure 5.8

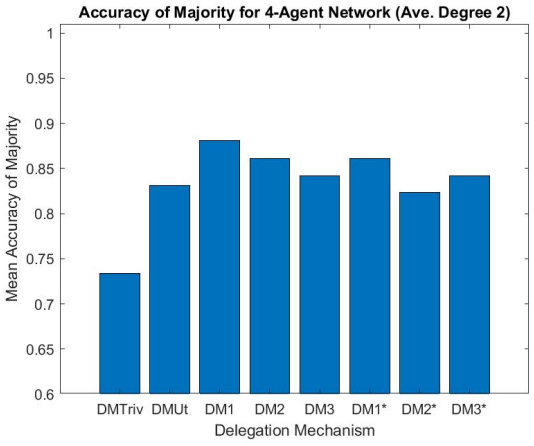


Figure 5.9

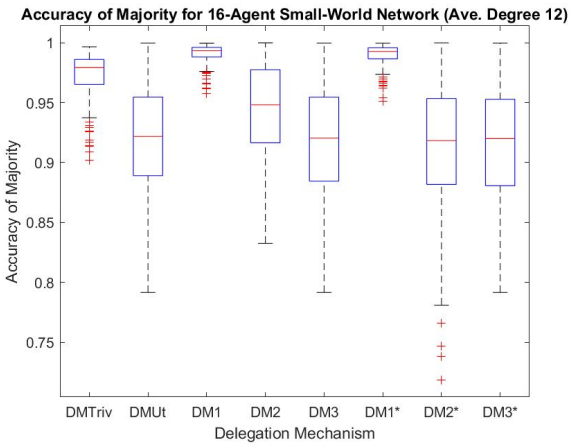


Figure 5.10

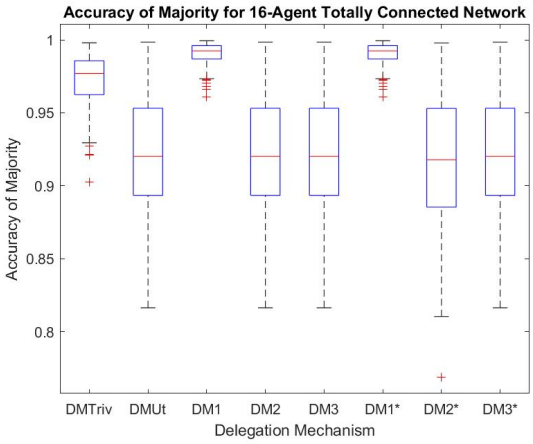


Figure 5.11

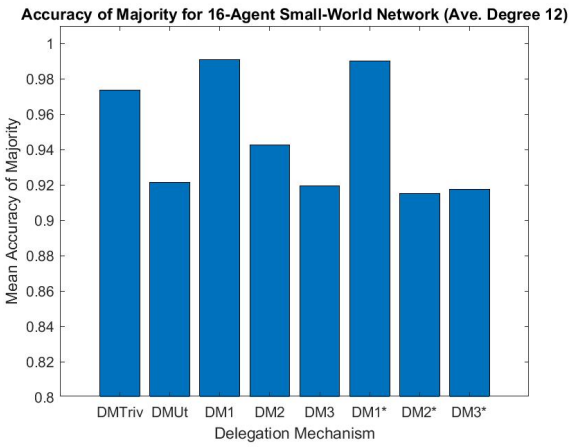
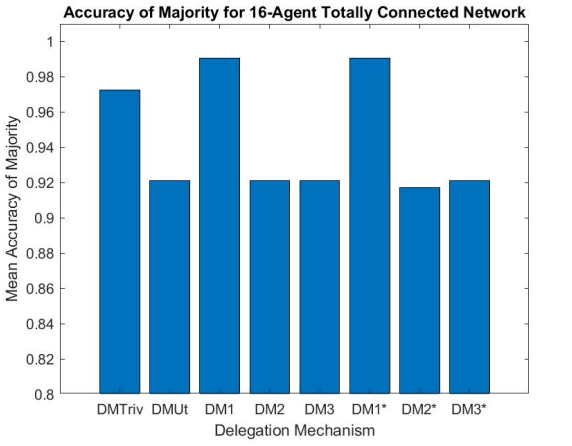


Figure 5.12



Chapter 6

Conclusions

This paper sought to establish delegation mechanisms under the partial delegation framework of liquid democracy and compare their effectiveness at improving the quality of a social decision. Three delegation mechanisms were motivated; the first has agents delegate their vote to all agents in their neighbourhood proportional to their accuracy, the second has agents delegate their vote proportional to the accuracy only to agents in their neighbourhood with accuracy greater than or equal to their own, and the final has agents delegate their vote proportional to the accuracy only to agents in their neighbourhood with accuracy greater than their own.

Following the development of these delegation mechanisms, we sought to investigate whether these mechanisms could be improved by the DeGroot learning process. Two different approaches to how the DeGroot learning process should be modelled in the liquid democracy context were motivated. The first approach has agents adjust their trust in other agents by reviewing their opinions even when two agents do not belong to each other's neighbourhoods. The second approach assumes agents have an awareness of the network structure; agents acknowledge that two agents in their neighbourhood may not belong to each other's neighbourhoods and adjust their opinions accordingly.

The result of these approaches on the delegation mechanisms was solved analytically. The process converged for all delegation mechanisms. For delegation mechanism 1, the first approach converged to a matrix whose elements could be calculated explicitly. Under the second approach, the delegation profile established by the delegation mechanism is invariant under the DeG-

root learning process. For a totally connected network, delegation mechanism 1 will remain invariant regardless of the approach to the DeGroot learning process. Under the first approach, delegation mechanism 2 saw agents award their full-weighted vote to those in their neighbourhood who began with the most trust in themselves regardless of their accuracy with the quirk that a vote is split among sinks proportional to their accuracy if an agent has multiple sinks in their neighbourhood. However, under the second approach, the mechanism reduced to the full delegation mechanism which maximises local utility. Regardless of approach, delegation mechanism 3 exhibits similar behaviour to this reduction, except agents with no sinks in their neighbourhood fail to delegate any weight of their vote so their vote does not get communicated to the voting mechanism.

With the delegation mechanisms and their DeGroot convergence matrices established, simulations were tested to analyse the quality of the decisions under these mechanisms over different network types and sizes. These were compared with the results for direct democracy and locally maximising utility. For $N = 4$ and $N = 16$, delegation mechanism 1 is consistently the optimal strategy across all network typologies and for all average degrees. Whether this trend continues for larger networks will need to be tested. It would seem that it would but because the delegation mechanism improves with greater network connectedness, it may be the case that it performs less well for large networks with a lower average degree.

Interestingly, delegation mechanism 1 performs the worst when it comes to maximising average utility across the agents in the network. This highlights the conflict between a motivation for the group and a motivation for the individual. The definition of utility explored is one determined by an agent's direct action; an agent seeks to have the best probability of having the full weight of their own vote communicated to the voting mechanism in favour of the ground truth. This is best achieved by delegating their vote to a guru with the highest accuracy. However, when there are a large number of voters, potential mistakes can average out through a phenomenon called the miracle of aggregation. If one guru has a high accuracy, they will become a sink for the agents around them and accumulate a large weight to communicate to the voting mechanism, reducing the number of gurus voting, and reducing the possibility of potential mistakes averaging out. Contrary to this, delegation mechanism 1 leads to agents sacrificing part of their vote

and delegating part of their vote to agents in their neighbourhood with a lower accuracy. This is counter-intuitive as a motivation for an individual agent, yet it seems to maximise the accuracy of a majority.

Direct democracy is known to improve a group decision as the number of agents increases through Condorcet's jury theorem. The simulations seems to suggest a similar pattern for delegation mechanism 1. The structure of the mechanism suggests an interesting direction for further research. All agents receive weight from their neighbours proportional to their accuracy. Consequently, since accuracies are assumed to be known, it would be interesting to compare the mechanism and direct democracy against a weighted direct vote; where each agent begins with weight proportional to their accuracy instead of every agent receiving a weight of 1.

Whereas delegation mechanism 1 exhibits the best performance, delegation mechanism 2 yields strong results comparable to direct democracy. What these delegation mechanisms have in common compared to all other delegation mechanisms examined is that these guarantee that every agent communicates to the voting mechanism to some extent. It seems that an important trait in ensuring a better chance of a majority support for the ground truth is to have as many agents as possible to vote, demonstrating the strength of the miracle of aggregation. Delegation mechanism 3, its DeGroot counterpart, and the DeGroot counterpart of delegation mechanism 2 reduce to the most accurate agents in the profile and this seems to worsen the accuracy of a majority. A further trend for these delegation mechanisms is they appear to worsen with greater connectedness of a network; this would be because it reduces the number of sinks and again means fewer mistakes can be lost through aggregation.

The DeGroot learning process was utilised under the pretense that a group would perform better if they used each other's opinion to inform their own. It was applied to all three delegation mechanisms and the convergence matrix from the process was shown analytically in each case. However, the DeGroot counterpart performed worse in the simulations for every delegation mechanism. The counterpart for delegation mechanism 1 was the second best performing mechanism; however, it performed worse than delegation mechanism 1 for the quality of the social choice in every case. Similarly, whereas delegation mechanism 2 performed strongly in simulations, its DeGroot coun-

terpart returned much worse results.

The success of delegation mechanism 1 is a positive result and it bears further research to determine if it can be improved and if it continues to be a effective strategy for larger networks. Much of the method for this paper is motivated by Bloembergen et al (2018) which focused on rationality; rationality is not discussed in this paper, so it would be interesting to introduce rationality constraints and to examine the effectiveness of delegation mechanism 1. Furthermore, this paper assumed homogeneity, so it would be fruitful to examine these delegation mechanisms in social choices where agents may have different ground truth given in either a deterministic or probabilistic profile. Finally, this paper assumed accuracy of an agent followed a normal distribution between the values 0.5 and 1; it would be interesting to see study in this framework which permits misinformation campaigns and false belief, allowing accuracy to take any value between 0 and 1.

Bibliography

Barabasi, Albert-Laszlo and Reka Albert (1999). "Emergence of Scaling in Random Networks". In: *Science* 286.5439, pp. 509–512. DOI: 10.1126/science.286.5439.509. eprint: http://www.sciencemag.org/cgi/reprint/286/5439/509.pdf. URL: http://www.sciencemag.org/cgi/content/abstract/286/5439/509.

Bloembergen, Daan, Davide Grossi, and Martin Lackner (2018). *On Rational Delegations in Liquid Democracy*. arXiv: 1802.08020 [cs.MA].

Blum, Christian and Christina Zuber (Aug. 2015). "Liquid Democracy: Potentials, Problems, and Perspectives". In: *Journal of Political Philosophy* 24. DOI: 10.1111/jopp.12065.

Boella, Guido et al. (Apr. 2018). "WeGovNow: A Map Based Platform to Engage the Local Civic Society". In: pp. 1215–1219. ISBN: 9781450356404. DOI: 10.1145/3184558.3191560.

Boldi, Paolo et al. (2009). *Viscous Democracy for Social Networks*.

Condorcet J.-A.-N. de Caritat, marquis de (1785). *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Paris: Imprimerie Royale.

DeGroot, Morris H. (1974). "Reaching a Consensus". In: *Journal of the American Statistical Association* 69.345, pp. 118–121. DOI: 10.1080/01621459.1974.10480137. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1974.10480137. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1974.10480137.

Erdös, P. and A. Rényi (1959). "On Random Graphs I". In: *Publicationes Mathematicae Debrecen* 6, p. 290.

George, Mathew (2020). *B-A Scale-Free Network Generation and Visualisation*. URL: https://www.mathworks.com/matlabcentral/fileexchange/11947-b-a-scale-free-network-generation-and-visualization.

Gölz, Paul et al. (2018). "The Fluid Mechanics of Liquid Democracy". In: *Lecture Notes in Computer Science*, pp. 188–202. ISSN: 1611-3349. DOI: 10.1007/978-3-030-04612-5_13. URL: http://dx.doi.org/10.1007/978-3-030-04612-5_13.

Kahng, A., S. Mackenzie, and A. D. Procaccia (2018). "Liquid Democracy: An Algorithmic Perspective". In: *AAAI 2018*. URL: https://par.nsf.gov/biblio/10063518.

Pundak, Golan (2020). *Random Regular generator*. URL: https://www.mathworks.com/matlabcentral/fileexchange/29786-random-regular-generator.

Watts, D.J. and S.H. Strogatz (1998). "Collective dynamics of 'small-world' networks". In: *Nature* 393, pp. 440–442.

Appendix A

accuracy.m

Listing A.1: accuracy.m

```
function [q]=accuracy(G)
% input: G - An NxN adjacency matrix of a graph
% output: q - An Nx1 vector of each agent's accuracy. Randomised by normal
% distribution N(0.75,0.1)

N=size(G);                          %Define the number of agents
a=0.1;                              %Set the variance
b=0.75;                             %Set the mean
q = a.*randn(N(1),1) + b;           %Randomise the accuracies
for i=1:N(1)                        %Fix to ensure all accuracies within range
    if q(i)>1
        q(i)=2-q(i);
    end
    if q(i)<0.5
        q(i)=1-q(i);
    end
end
end
```

Appendix B

partdel.m

Listing B.1: partdel.m

```
function [D1]=partdel(D)
% input: D - An NxN delegation profile
% output: D1 - An NxN path profile for partial delegations along
     all paths

sz=size(D);
D1=zeros(sz(1));     %Preset D1 for efficiency
for n=1:sz(1)-1      %Adds the path profiles of all path lenghts
    D1=D1+D;
    D=omega(D);
end
end
```

Appendix C

omega.m

Listing C.1: omega.m

```
function [B]=omega(A)
% input: A - An NxN path profile
% output: B - The path profile of +1 length to path profile A

A2=A-diag(diag(A));      %Removes diagonal elements for first
    iterations
B=A2*A2;
B=B-diag(diag(B));       %Removes any new diagonal elements
end
```

Appendix D

vote.m

Listing D.1: vote.m

```
function [D2]=vote(D1)
% input: D1 - An NxN path profile of all path lengths
% output: D2 - An NxN guru profile

sz=size(D1);
D2=D1;                      %Sets diagonal elements to retained
    delegations
for i=1:sz(1)               %Weights paths by agents' retained
    delegations
    for j=1:sz(2)
        if i~=j
           D2(i,j)=D1(i,j)*D1(j,j);
        end
    end
end
end
```

Appendix E

utility.m

Listing E.1: utility.m

```
function [u]=utility(D,q)
% input: D - An NxN delegation profile
%        q - An Nx1 accuracy vector
% output: u - An Nx1 utility vector

D1=partdel(D); %calculate path profile of all lengths
D2=vote(D1);   %calculate guru profile
u=D2*q;        %sum weights at gurus multiplied by accuracies
end
```

Appendix F

DelMech1.m

Listing F.1: DelMech1.m

```
function [D]=DelMech1(G,q)
% input: G - An NxN adjacency matrix of a graph
%        q - An Nx1 accuracy vector
% output: D - An NxN delegation profile

N=size(G);              %Set number of agents
v=ones(N(1),1);
G=G+diag(v);            %Ensure adjacency matrix is reflexive
for i=1:N(1)            %Delegate to each agent proportinal to
    accuracy.
   D(i,:)=(G(i,:).*q')/sum(G(i,:).*q');
end
end
```

Appendix G

DelMech2.m

Listing G.1: DelMech2.m

```
function [D]=DelMech2(G,q)
% input: G - An NxN adjacency matrix of a graph
%        q - An Nx1 accuracy vector
% output: D - An NxN delegation profile

N=size(G);              %Set number of agents
v=ones(N(1),1);
G=G+diag(v);            %Ensure adjacency matrix is reflexive
for i=1:N(1)            %Create matrix of delegations proportional
    to accuracy
   M(i,:)=(G(i,:).*q')/sum(G(i,:).*q');
end
for i=1:N(1)            %Restrict delegations to those the same or
    higher
        for j=1:N(1)
            if M(i,j)<M(i,i)
                G(i,j)=0;
            end
        end
end
for i=1:N(1)            %Renormalise delegations
   D(i,:)=(G(i,:).*q')/sum(G(i,:).*q');
end
end
```

Appendix H

DelMech3.m

Listing H.1: DelMech3.m

```
function [D]=DelMech3(G,q)
% input: G - An NxN adjacency matrix of a graph
%        q - An Nx1 accuracy vector
% output: D - An NxN delegation profile

N=size(G);          %Set number of agents
v=ones(N(1),1);
G=G+diag(v);        %Ensure adjacency matrix is reflexive
for i=1:N(1)        %Create matrix of delegations proportional
    to accuracy
   M(i,:)=(G(i,:).*q')/sum(G(i,:).*q');
end
ZerosTest=0;
for i=1:N(1)        %Restrict delegations to those the same
    higher.
        for j=1:N(1)
            if M(i,j)<=M(i,i)
                G(i,j)=0;
                ZerosTest=ZerosTest+1;
            end
        end
        if ZerosTest==N(1)   %if no agent with higher acc, self-
    delegate
            G(i,i)=1;
        end
        ZerosTest=0;
end
for i=1:N(1)        %Renormalise delegations
```

```
26      D(i,:)=(G(i,:).*q')/sum(G(i,:).*q');
27  end
28  end
```

Appendix I

DelMechTriv.m

Listing I.1: DelMechTriv.m

```
function [D]=DelMechTriv(G)
% input: G - An NxN adjacency matrix of a graph
% output: D - An NxN delegation profile

N=size(G);          %Set number of agents
v=ones(N(1),1);
D=diag(v);          %Create profile with diagonal of 1s
end
```

Appendix J

CDelMechUt.m

Listing J.1: DelMechUt.m

```
function [D]=DelMechUt(G,q)
% input: G - An NxN adjacency matrix of a graph
%        q - An Nx1 accuracy vector
% output: D - An NxN delegation profile

N=size(G);          %Set number of agents
v=ones(N(1),1);
G=G+diag(v);        %Ensure adjacency matrix is reflexive
for i=1:N(1)        %Create matrix of delegations proportional to
    accuracy
   M(i,:)=G(i,:).*q';
end
for i=1:N(1)        %Restrict delegations to only max accuracy
        for j=1:N(1)
            if M(i,j)==max(M(i,:))
                G(i,j)=1;
            else
                G(i,j)=0;
            end
        end
end
for i=1:N(1)        %Renormalise delegations
   D(i,:)=(G(i,:).*q')/sum(G(i,:).*q');
end
end
```

Appendix K

DelMech1Deg.m

Listing K.1: DelMech1Deg.m

```
function [D]=DelMech1Deg(G,q)
% input: G - An NxN adjacency matrix of a graph
%        q - An Nx1 accuracy vector
% output: D - An NxN delegation profile

N=size(G);              %Set number of agents
v=ones(N(1),1);
G=G+diag(v);            %Ensure adjacency matrix is reflexive
D=zeros(N(1),N(2));
M=zeros(N(1),N(2));
C=zeros(N(1),N(2));
for i=1:N(1)            %Delegate to each agent proportinal to
    accuracy.
   M(i,:)=(G(i,:).*q')/sum(G(i,:).*q');
end
for i=1:N(1)            %matrix of weightings relative to their
    neighbourhood
    for j=1:N(1)
        C(i,j)=sum(M(i,:).*G(j,:))*G(i,j);
    end
end
for i=1:N(1)            %matrix of converged elements
    for j=1:N(1)
        D(i,j)=sum(M(:,j).*G(i,:)'*C(i,j))/sum(sum(M.*G(i,:)').*
    C(i,:));
    end
end
end
```

Appendix L

DelMech2Deg.m

Listing L.1: DelMech2Deg.m

```
function [D]=DelMech2Deg(G,q)
% input: G - An NxN adjacency matrix of a graph
%        q - An Nx1 accuracy vector
% output: D - An NxN delegation profile

N=size(G);          %Set number of agents
v=ones(N(1),1);
D1=DelMech2(G,q);
G=G+diag(v);        %Ensure adjacency matrix is reflexive
M=zeros(N);
D=zeros(N);
for i=1:N(1)        %Matrix of agents' trust in themselves
    for j=1:N(1)
        if G(i,j)==1
            M(i,j)=D1(j,j);
        end
    end
end
for i=1:N(1)       %Delegate vote to agent with most self-trust
    for j=1:N(1)
        if M(i,j)==max(M(i,:))
            D(i,j)=1;
        else
            D(i,j)=0;
        end
    end
end
for i=1:N(1)      %proportional convergence for multi-sinks
```

```
29      if sum(D(i,:))>1
30     D(i,:)=(D(i,:).*q')/sum(D(i,:).*q');
31      end
32 end
33 end
```

Appendix M

DelMech3Deg.m

Listing M.1: DelMech3Deg.m

```
function [D]=DelMech3Deg(G,q)
% input: G - An NxN adjacency matrix of a graph
%        q - An Nx1 accuracy vector
% output: D - An NxN delegation profile

N=size(G);          %Set number of agents
v=ones(N(1),1);
G=G+diag(v);        %Ensure adjacency matrix is reflexive
D=zeros(N);
M=zeros(N);
for i=1:N(1)        %Create matrix of delegations proportional to
    accuracy
   M(i,:)=(G(i,:).*q')/sum(G(i,:).*q');
end
ZerosTest=0;
for i=1:N(1)        %Create adjacency graph of only higher
   accuracy
        for j=1:N(1)
            if M(i,j)<=M(i,i)
                G(i,j)=0;
                ZerosTest=ZerosTest+1;
            end
        end
        if ZerosTest==N(1)
            G(i,i)=1;
        end
        ZerosTest=0;
end
```

```
for i=1:N(1)        %neighbours of sinks to delegate to sink
    for j=1:N(1)
        if G(i,j)==1
            if G(j,j)==1
                G(i,j)=1;
            else
                G(i,j)=0;
            end
        end
    end
end
for i=1:N(1)        %renormalise delegations
    if sum(G(i,:))>0
   D(i,:)=(G(i,:).*q')/sum(G(i,:).*q');
    end
end
end
```

Appendix N

MajProb.m

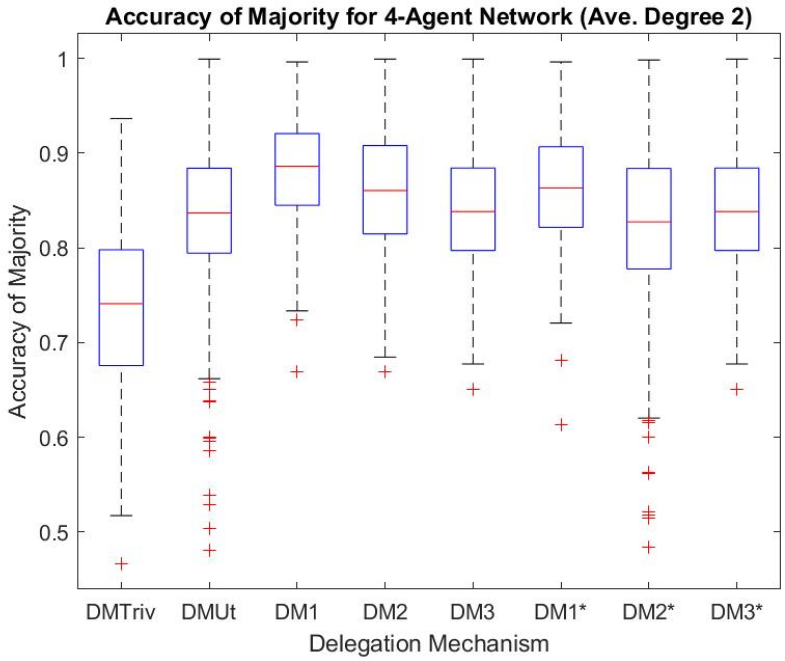
Listing N.1: MajProb.m

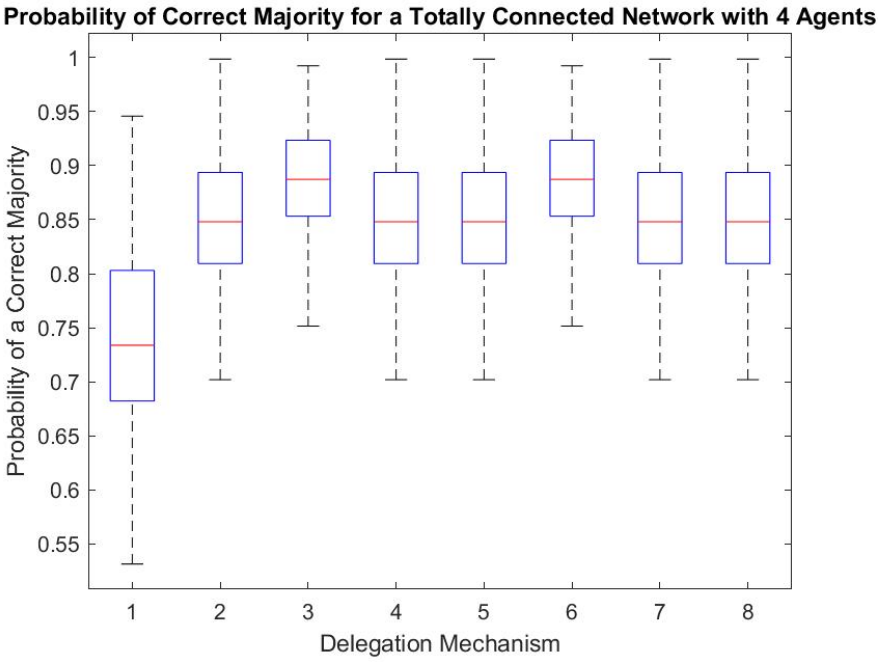
```
function [P]=MajProb(D,q)
% input: D - An NxN delegation profile
%        q - An Nx1 accuracy vector
% output: P - A 1x1 probability that the ground truth has a
    majority

D1=partdel(D);              %create path profile
D2=vote(D1);                %create guru profile
N=size(D2);
P=0;
weight=zeros(N(1),1);
for i=1:N(1)                %calculate weight vector from guru
    profile
    weight(i)=sum(D2(:,i));
end
threshold=sum(weight)/2;    %threshold is half the overall weight
Removed=0;
for i=1:N(1)                %remove accuracies for agents with
    weight 0
    if weight(i)==0
        q(i-Removed)=[];
        Removed=Removed+1;
    end
end
weight=nonzeros(weight);    %remove 0s from weight vector
N2=size(weight);
Iter=2^N2(1)-1;
qstep=q;
```

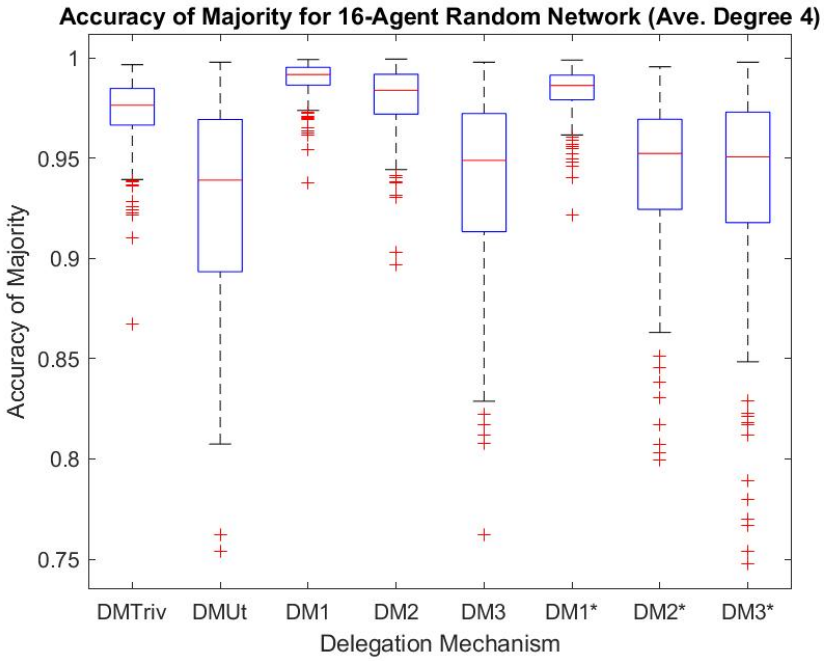
```
while Iter>0                %iterate through every guru vote
    combination
     IterBin=sparse(flip(de2bi(Iter)));
     if size(IterBin, 2)<N2(1)   %convert iteration step to
    binary vector
          addend=zeros(1,N2(1)-size(IterBin, 2));
          IterBin=sparse([addend IterBin]);
     end
     if IterBin*weight>threshold %if majority for ground truth,
    add prob
          for i=1:N2                    %subtract accuracy from 1 if
     vote incorrect
               if IterBin(i)==0
                    qstep(i)=1-qstep(i);
               end
          end
          prob=prod(qstep);             %calculate probability of
    vote combo
          P=P+prob;
          qstep=q;                      %reset accuracies
     end
     Iter=Iter-1;                  %prepare for next vote
    combination
end
end
```

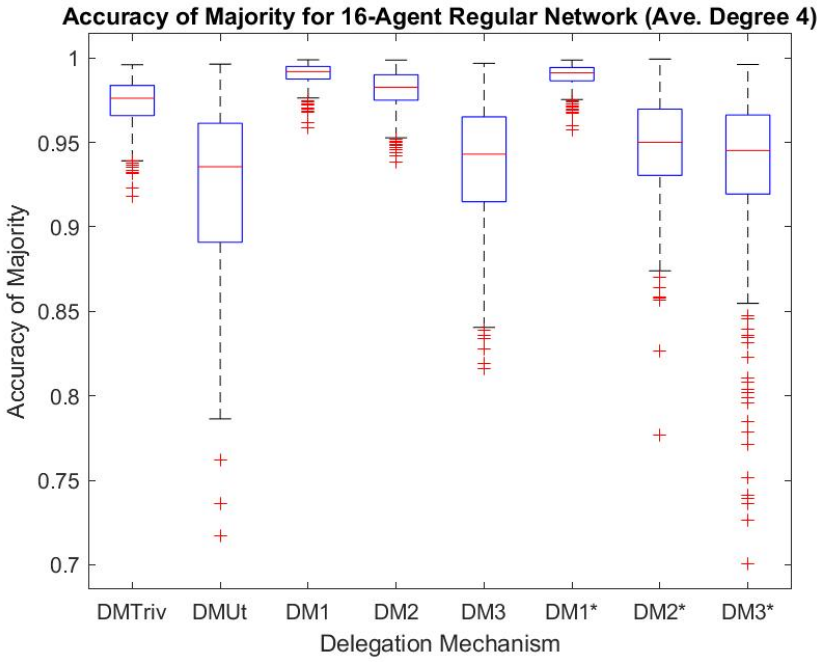
Appendix O

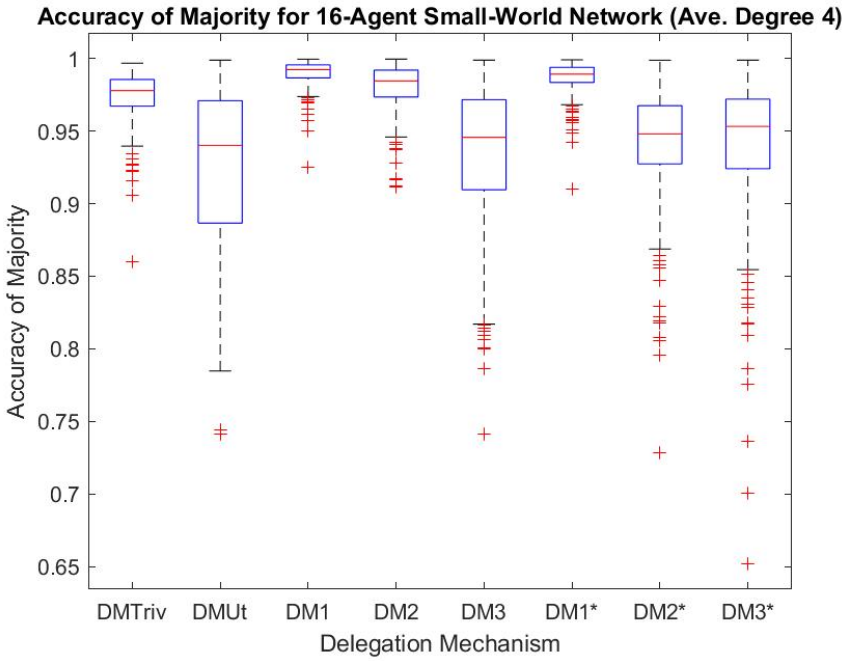
Probability of Majority Box Plots

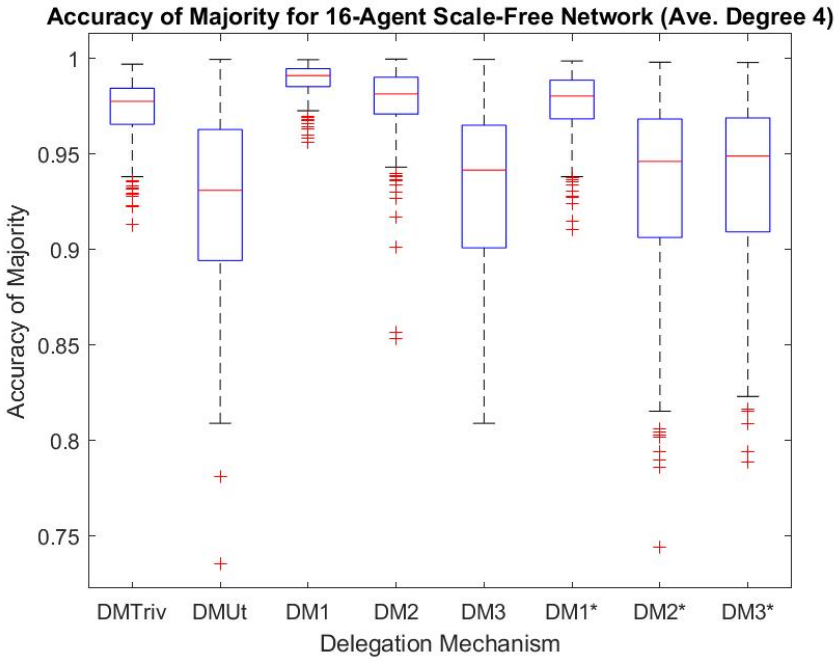


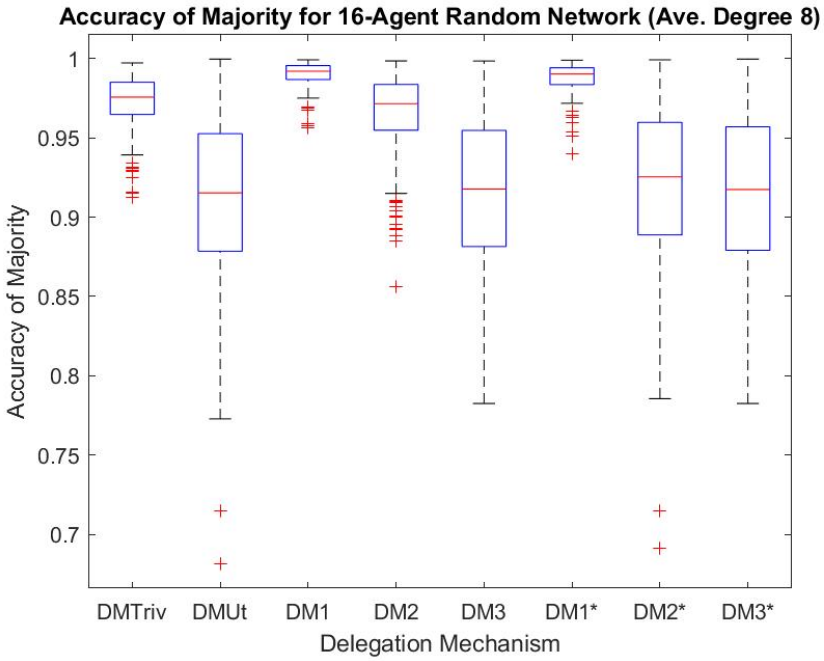

Probability of Correct Majority for a Totally Connected Network with 4 Agents
Probability of a Correct Majority
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
1
2
3
4
5
6
7
8
Delegation Mechanism

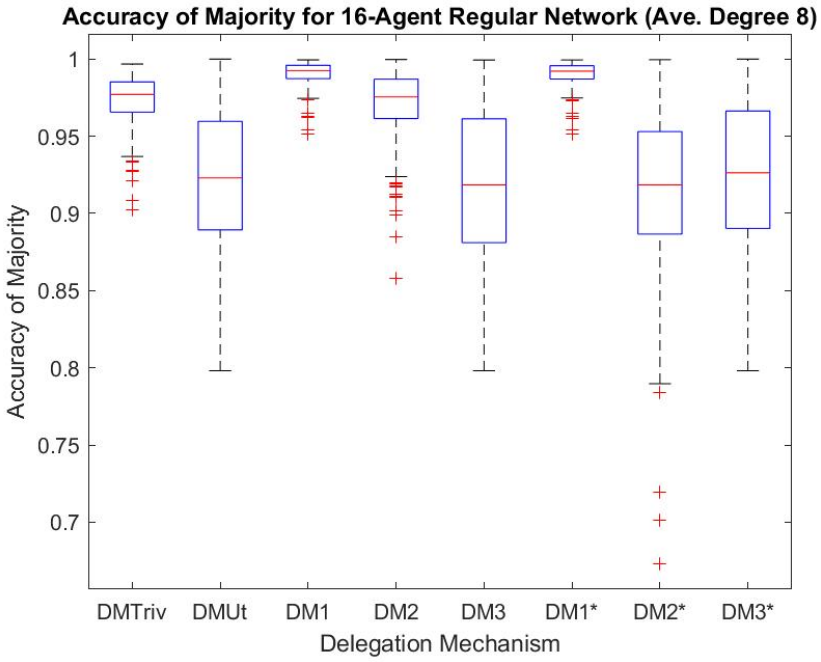

Accuracy of Majority for 16-Agent Random Network (Ave. Degree 4)
Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

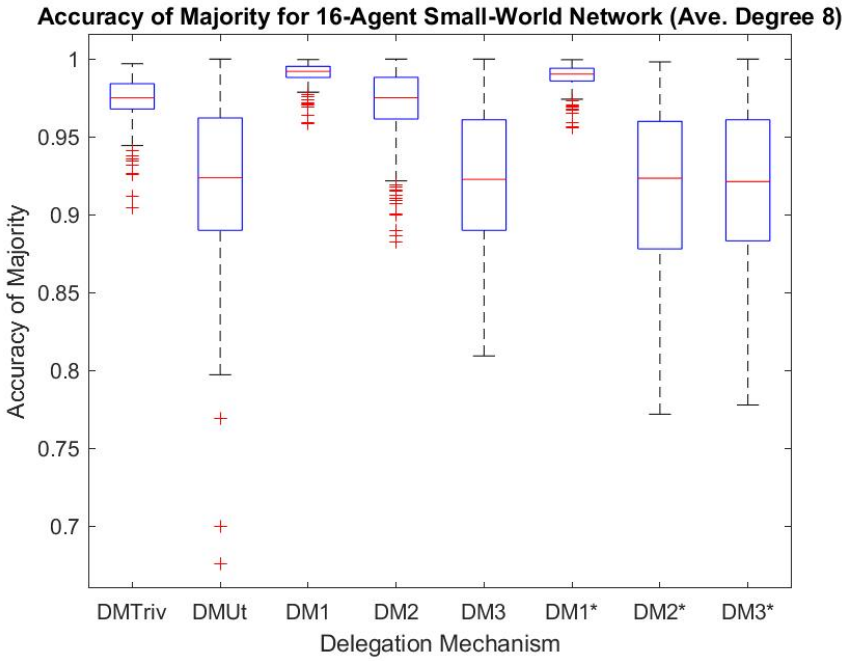

Accuracy of Majority for 16-Agent Regular Network (Ave. Degree 4)
Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
0.7
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

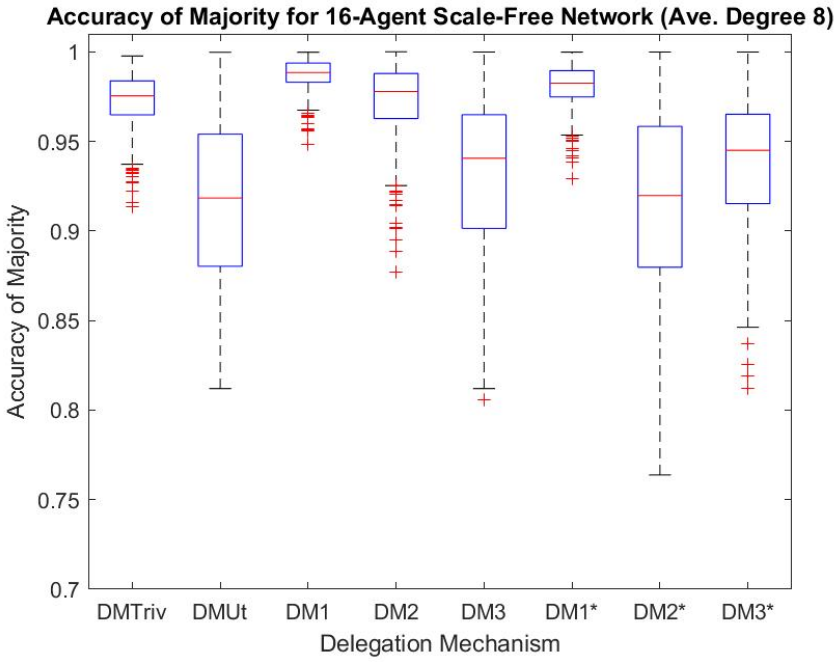

Accuracy of Majority for 16-Agent Small-World Network (Ave. Degree 4)
Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

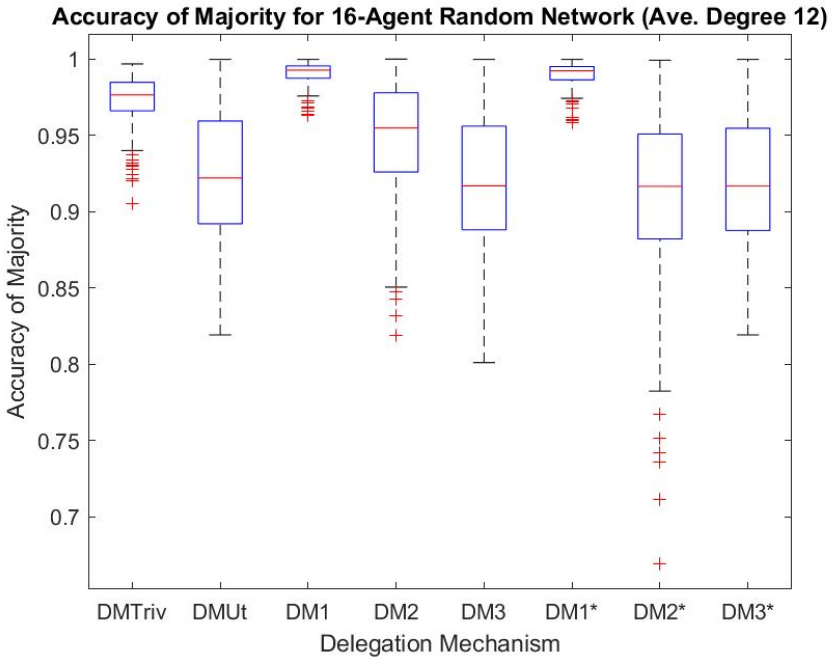

Accuracy of Majority for 16-Agent Scale-Free Network (Ave. Degree 4)
Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

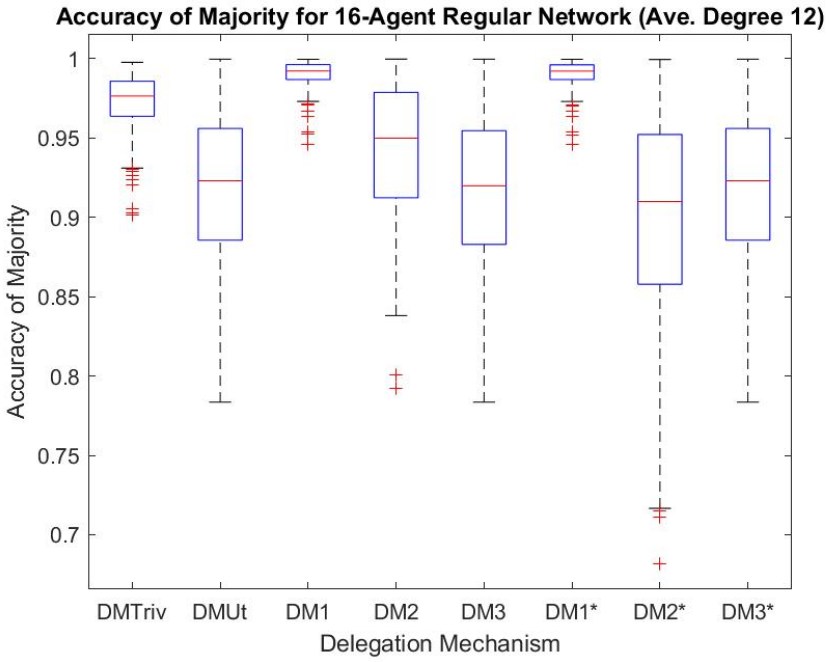

Accuracy of Majority for 16-Agent Random Network (Ave. Degree 8)
Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
0.7
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

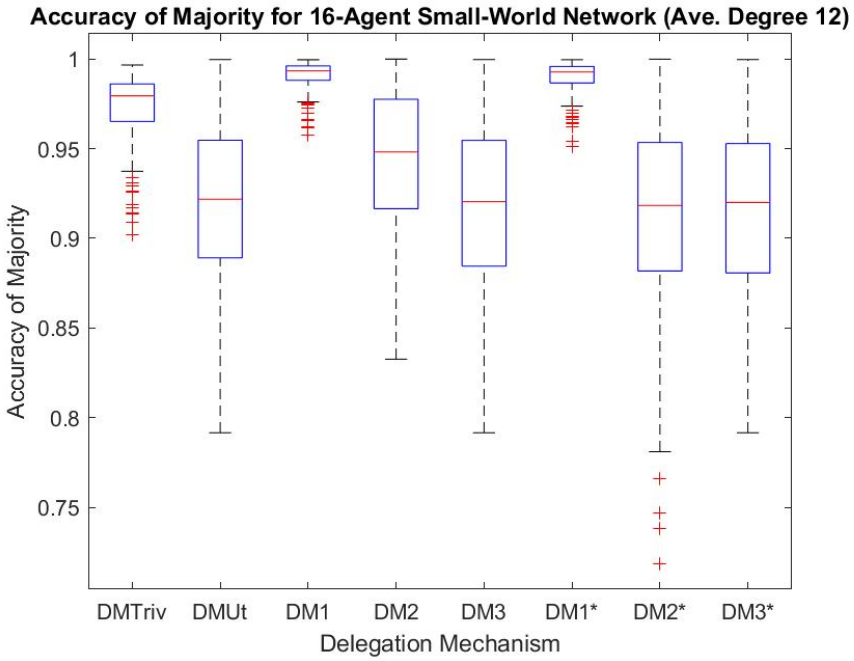

Accuracy of Majority for 16-Agent Regular Network (Ave. Degree 8)
Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
0.7
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

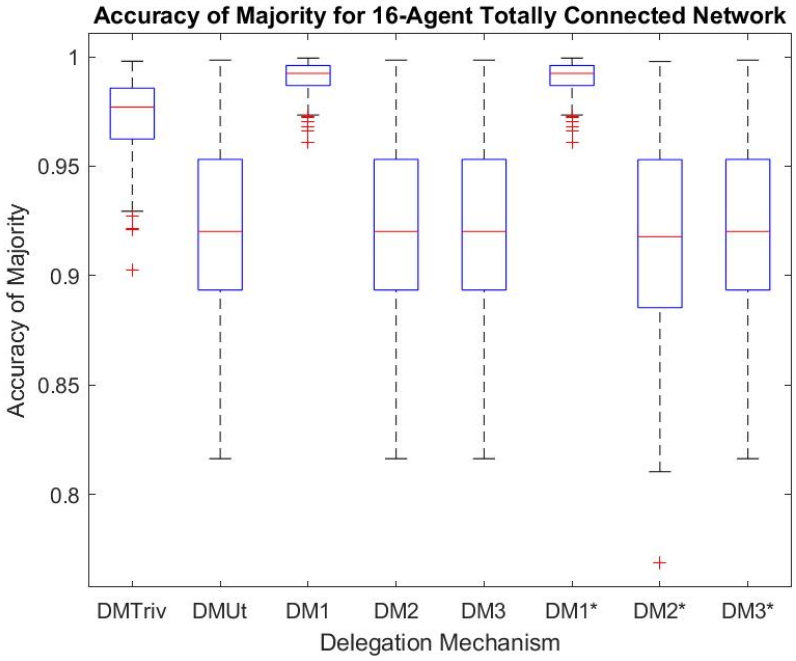

Accuracy of Majority for 16-Agent Small-World Network (Ave. Degree 8)
Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
0.7
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism


Accuracy of Majority for 16-Agent Scale-Free Network (Ave. Degree 8)
Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
0.7
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism


Accuracy of Majority for 16-Agent Random Network (Ave. Degree 12)
Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
0.7
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

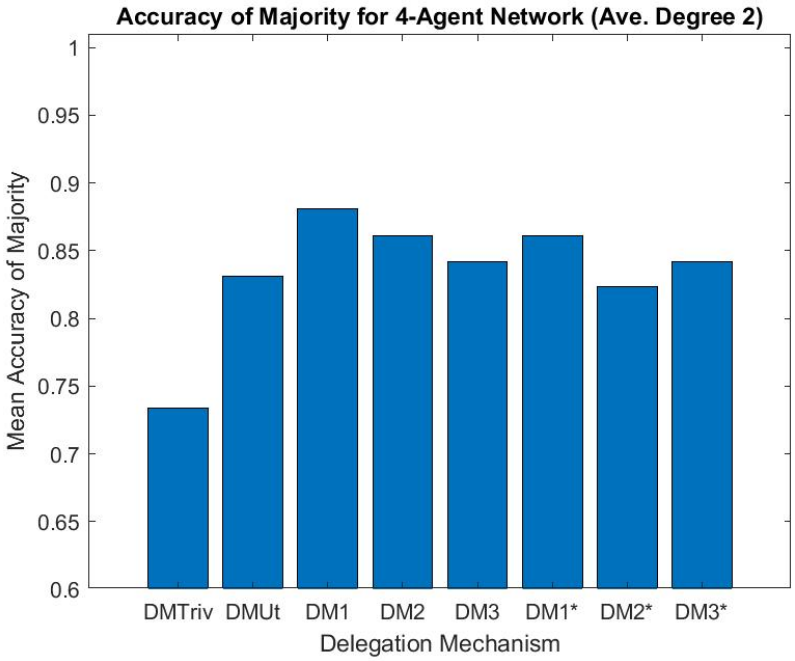

Accuracy of Majority for 16-Agent Regular Network (Ave. Degree 12)
Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
0.7
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

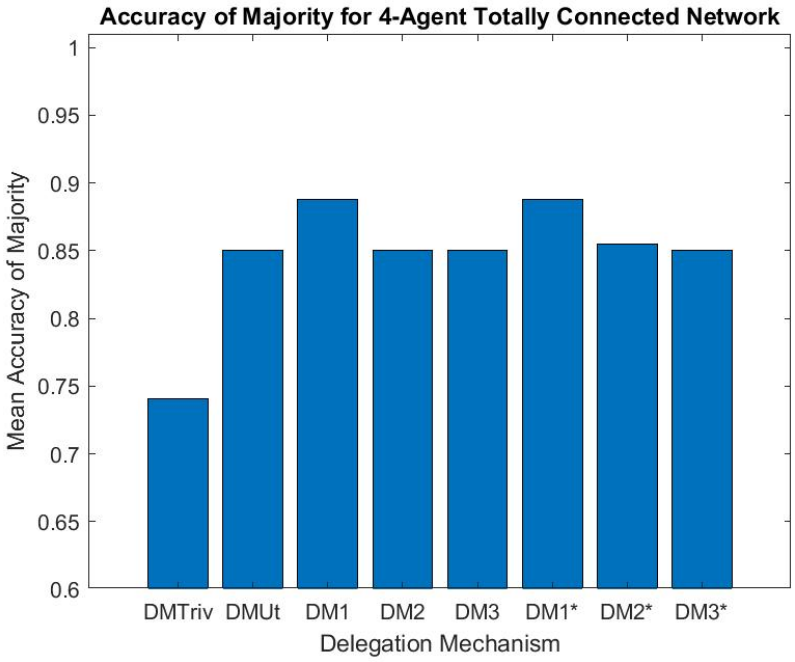

Accuracy of Majority for 16-Agent Small-World Network (Ave. Degree 12)
Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

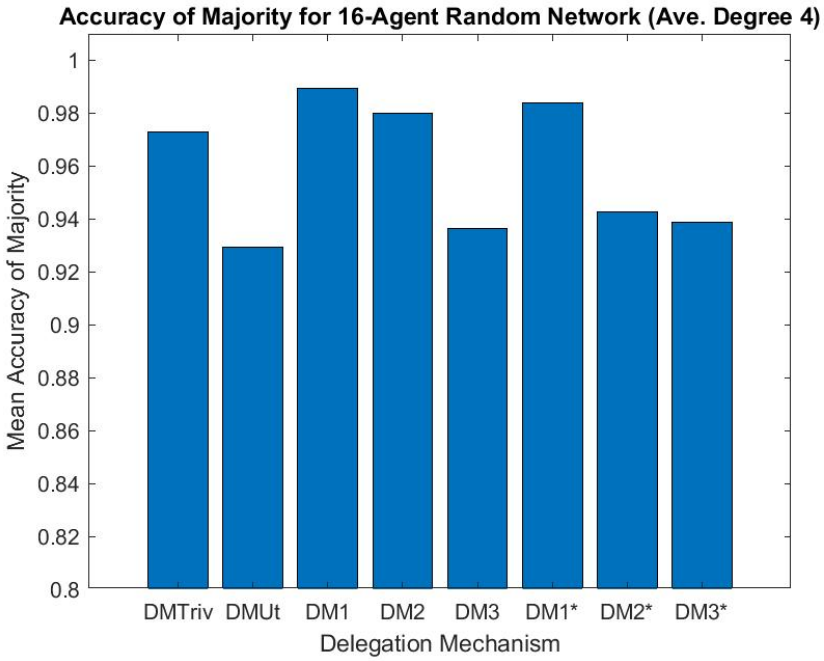

Accuracy of Majority for 16-Agent Totally Connected Network
Accuracy of Majority
1
0.95
0.9
0.85
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

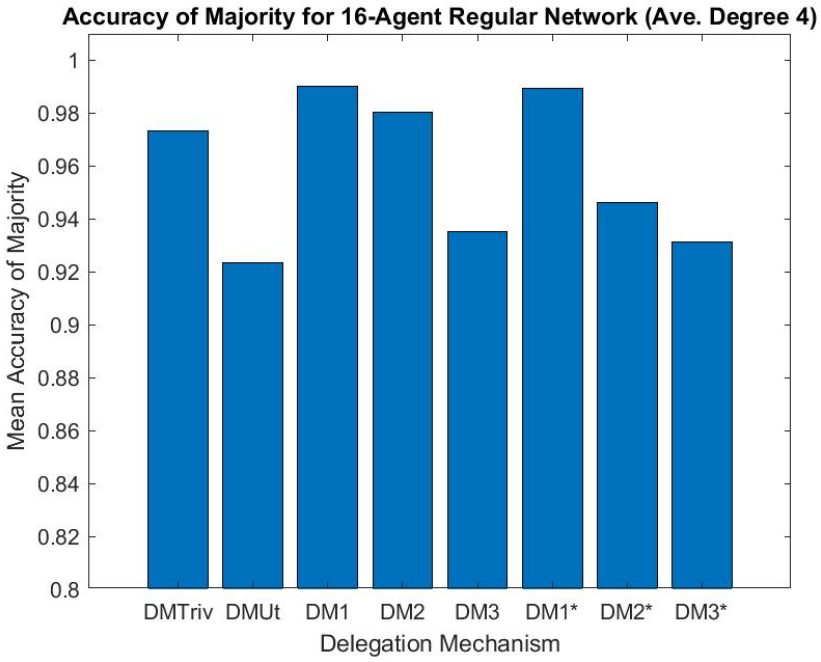
Appendix P

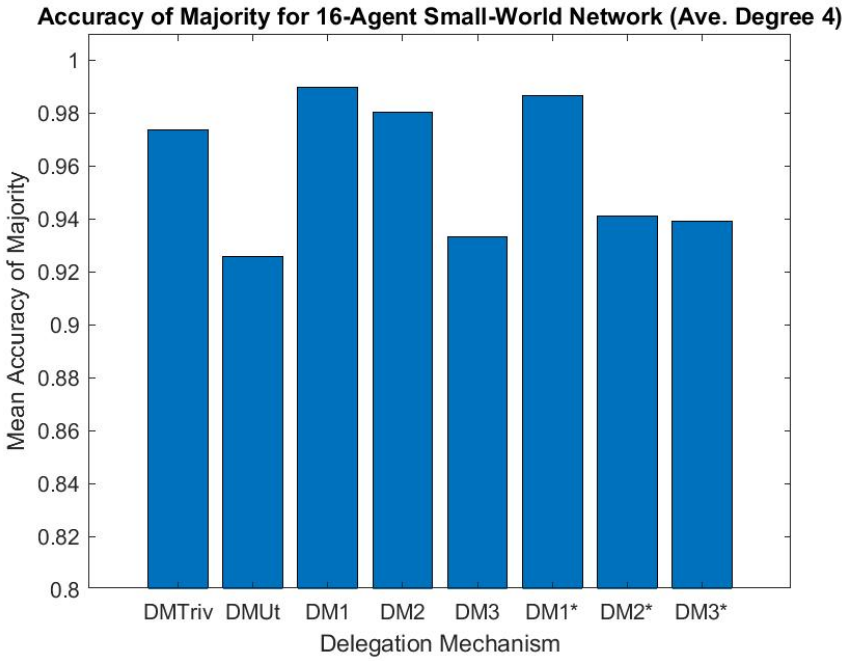
Probability of Majority Bar Charts

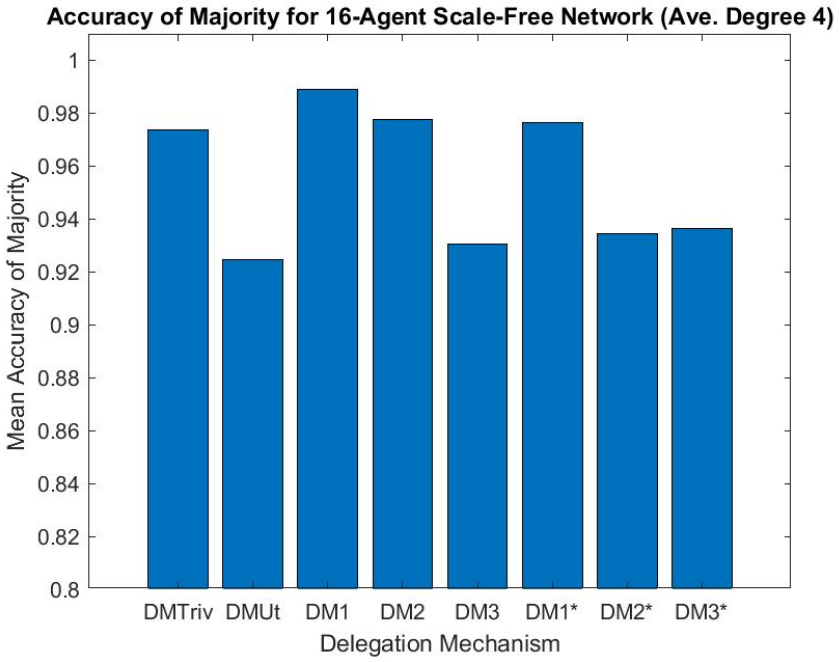


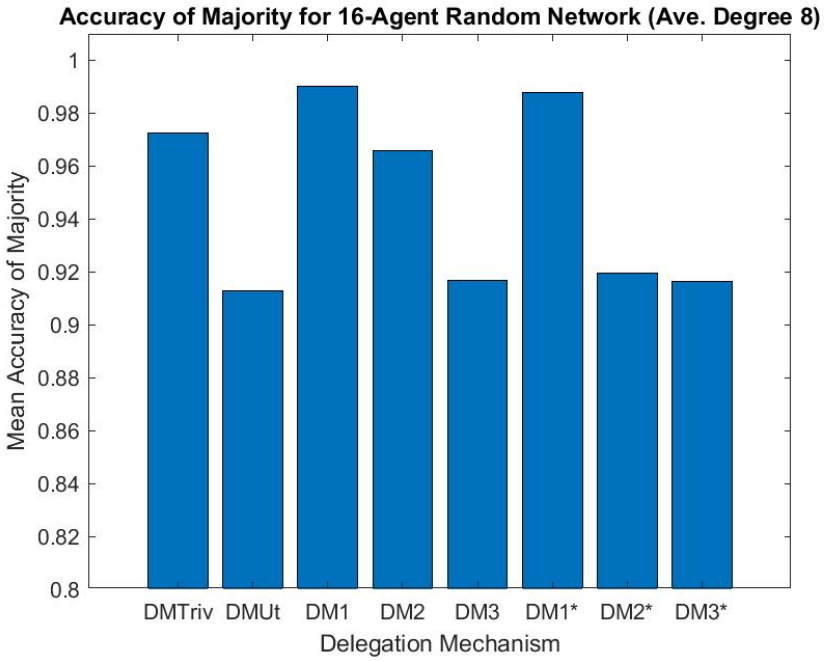

Accuracy of Majority for 4-Agent Totally Connected Network
Mean Accuracy of Majority
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

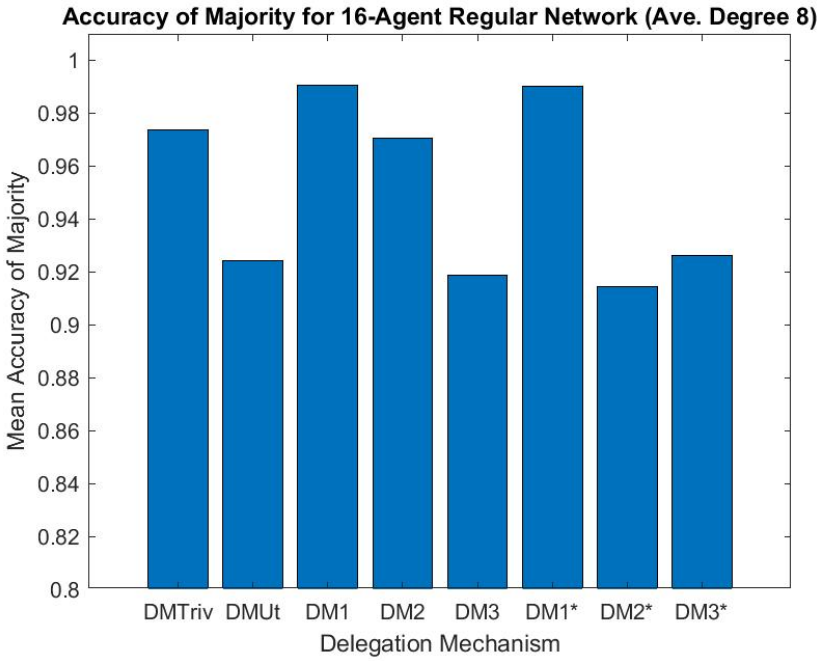

Accuracy of Majority for 16-Agent Random Network (Ave. Degree 4)
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

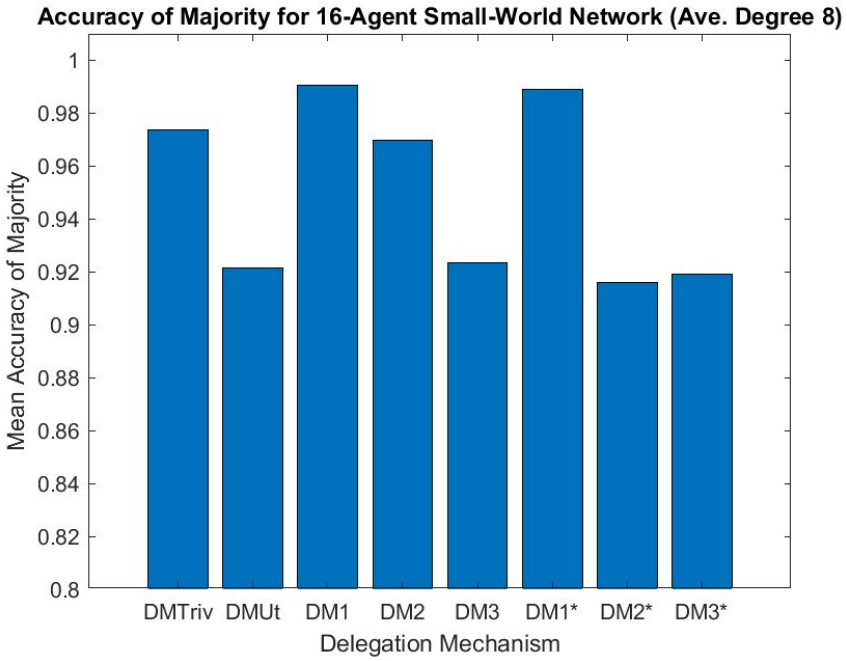

Accuracy of Majority for 16-Agent Regular Network (Ave. Degree 4)
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

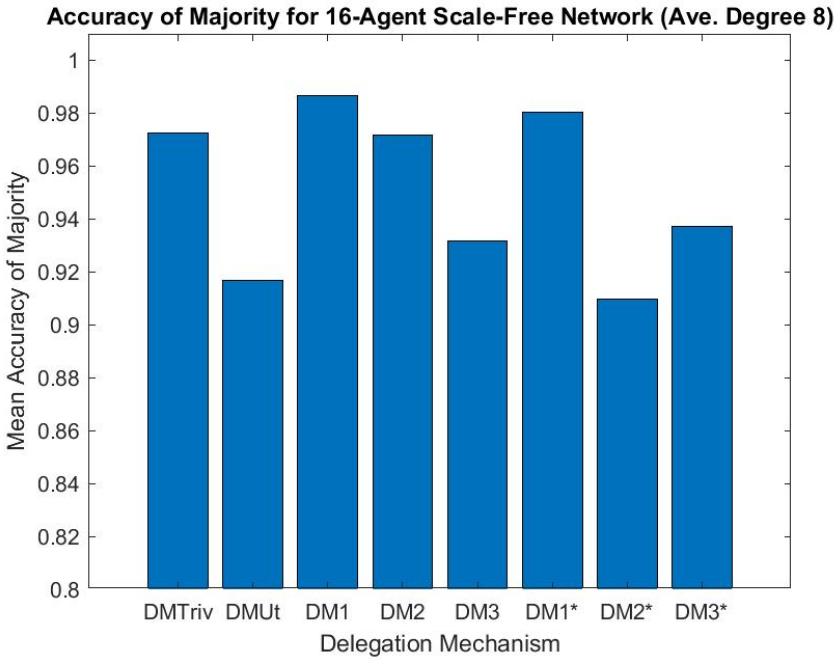

Accuracy of Majority for 16-Agent Small-World Network (Ave. Degree 4)
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

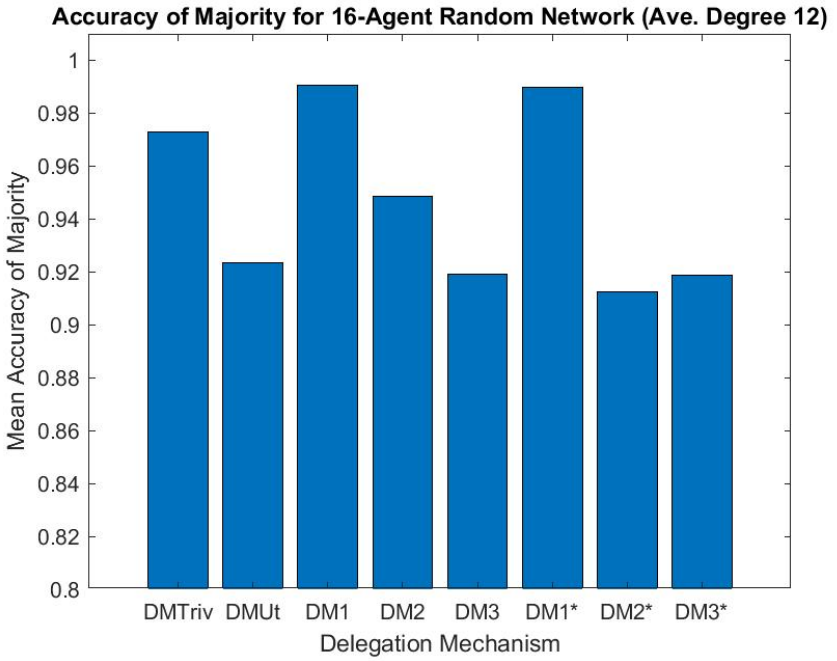

Accuracy of Majority for 16-Agent Scale-Free Network (Ave. Degree 4)
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

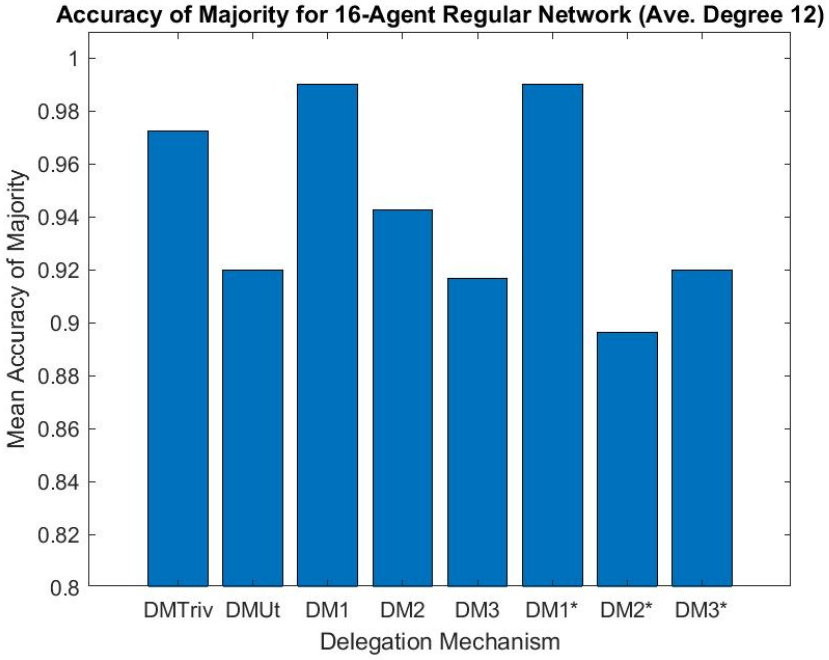

Accuracy of Majority for 16-Agent Random Network (Ave. Degree 8)
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

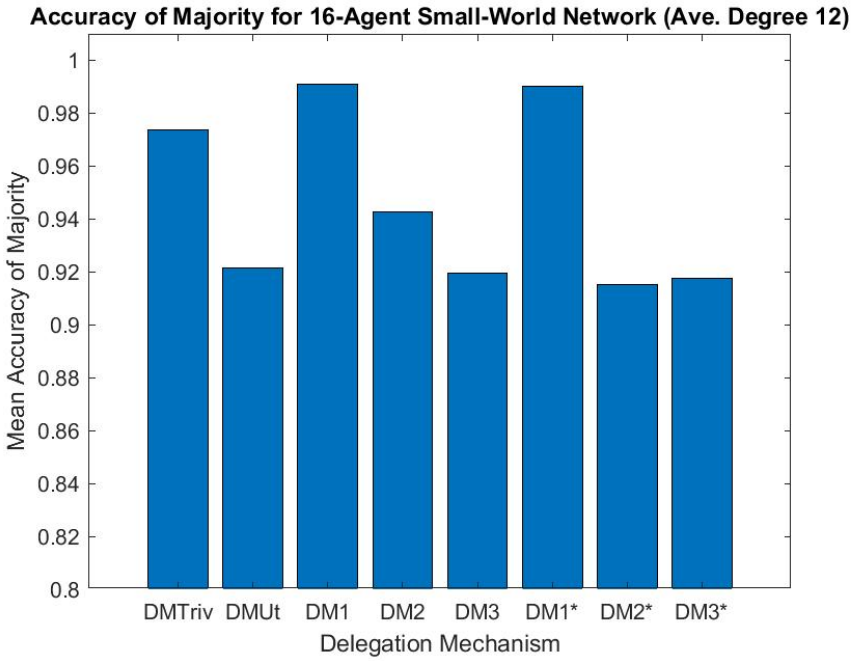

Accuracy of Majority for 16-Agent Regular Network (Ave. Degree 8)
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

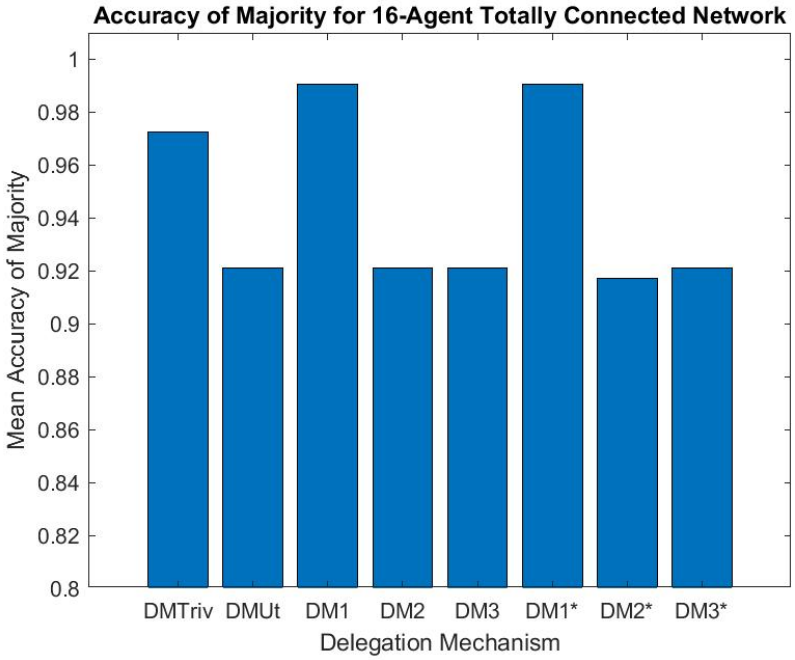

Accuracy of Majority for 16-Agent Small-World Network (Ave. Degree 8)
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism


Accuracy of Majority for 16-Agent Scale-Free Network (Ave. Degree 8)
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism


Accuracy of Majority for 16-Agent Random Network (Ave. Degree 12)
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

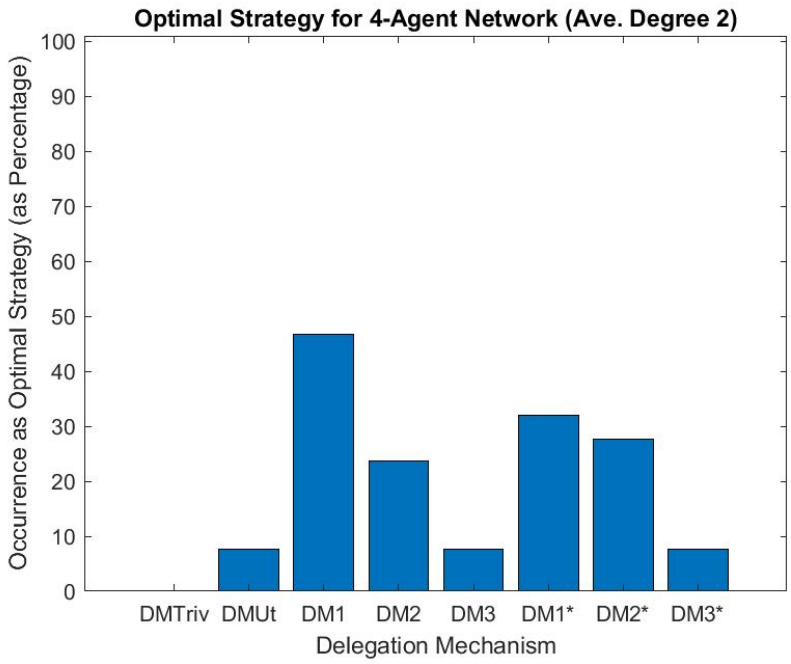

Accuracy of Majority for 16-Agent Regular Network (Ave. Degree 12)
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

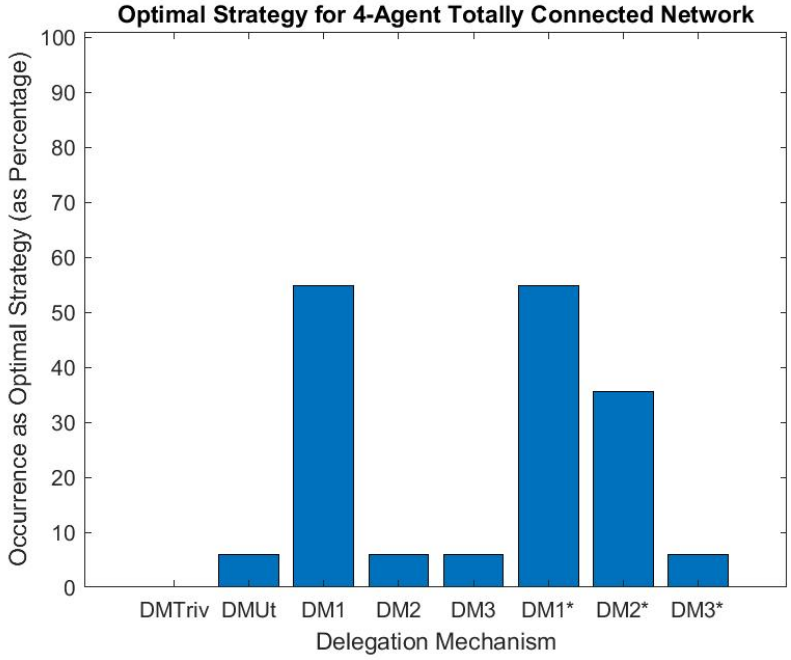

Accuracy of Majority for 16-Agent Small-World Network (Ave. Degree 12)
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

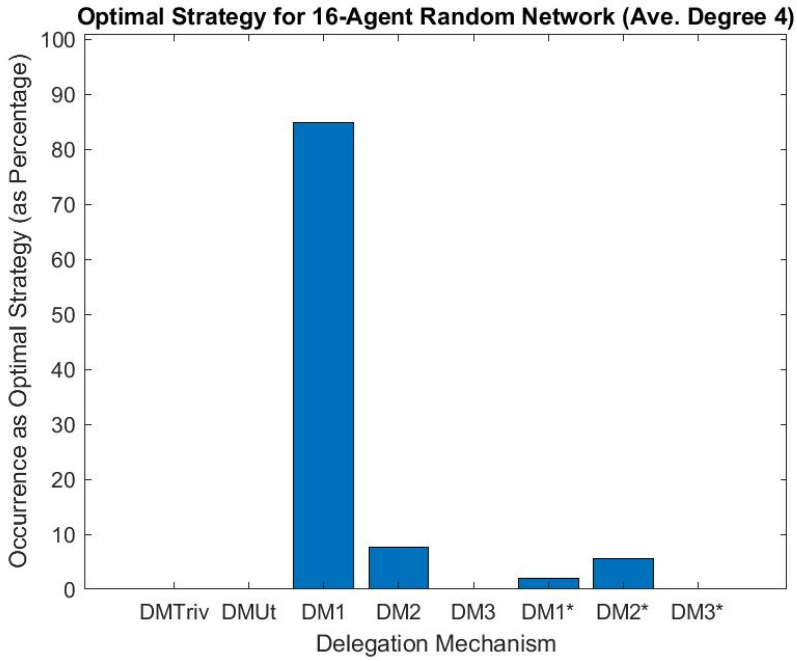

Accuracy of Majority for 16-Agent Totally Connected Network
Mean Accuracy of Majority
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

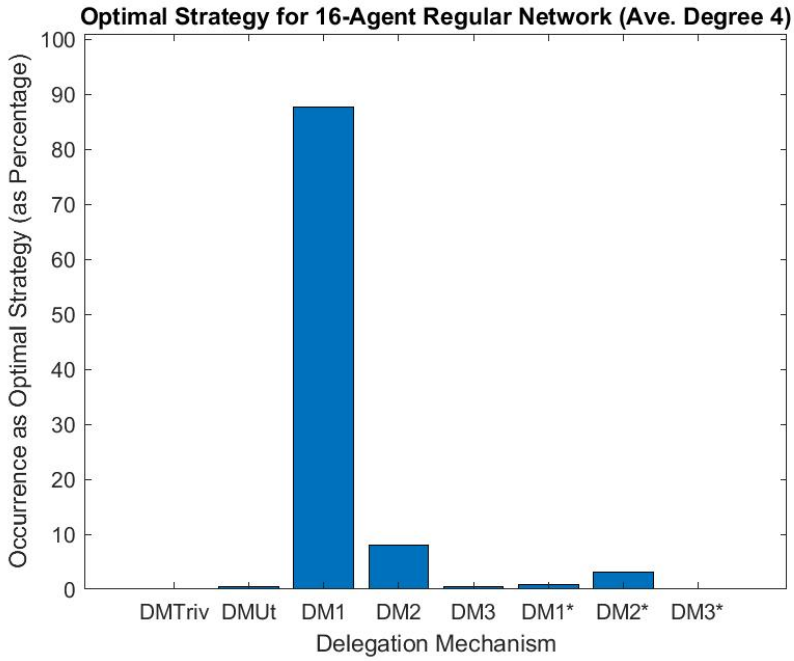
Appendix Q

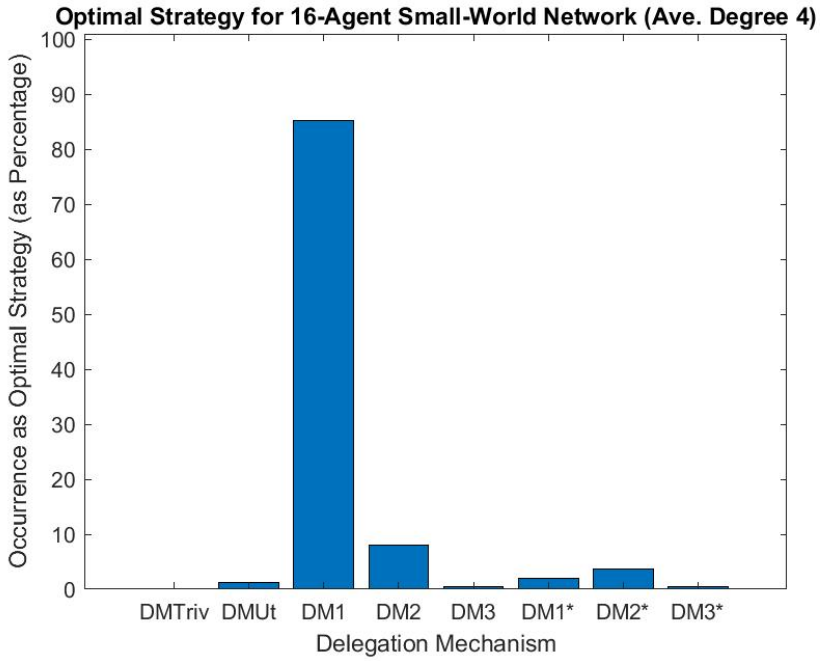
Optimal Strategy Bar Charts

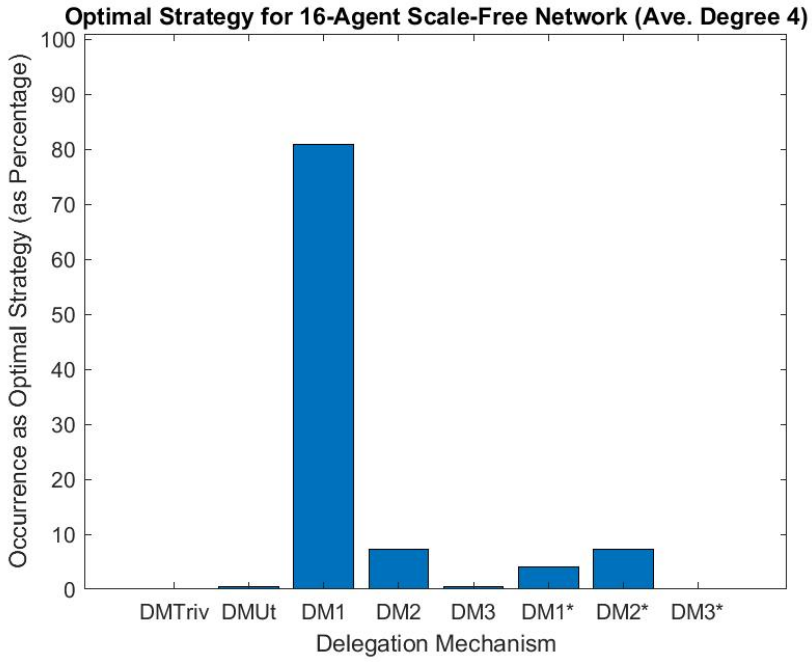


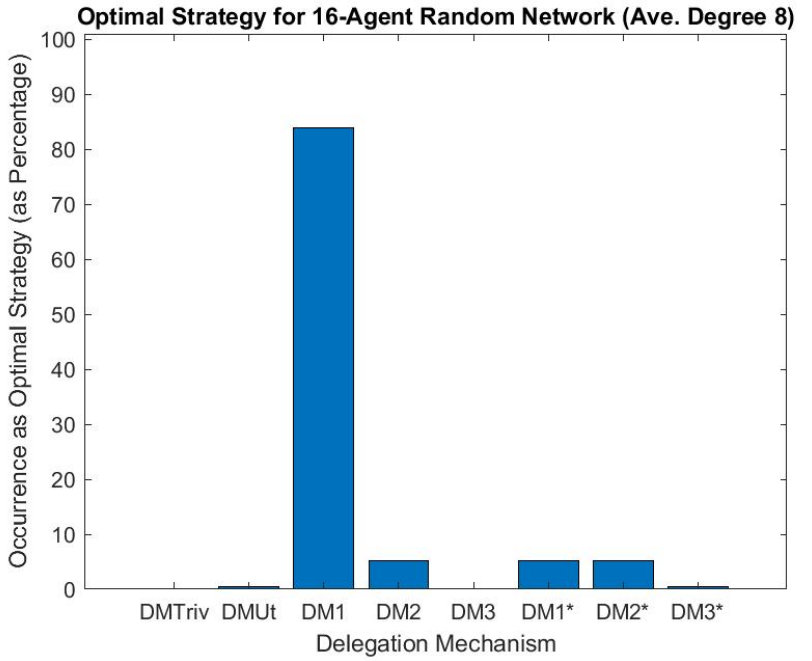

Optimal Strategy for 4-Agent Totally Connected Network
Occurrence as Optimal Strategy (as Percentage)
0
10
20
30
40
50
60
70
80
90
100
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

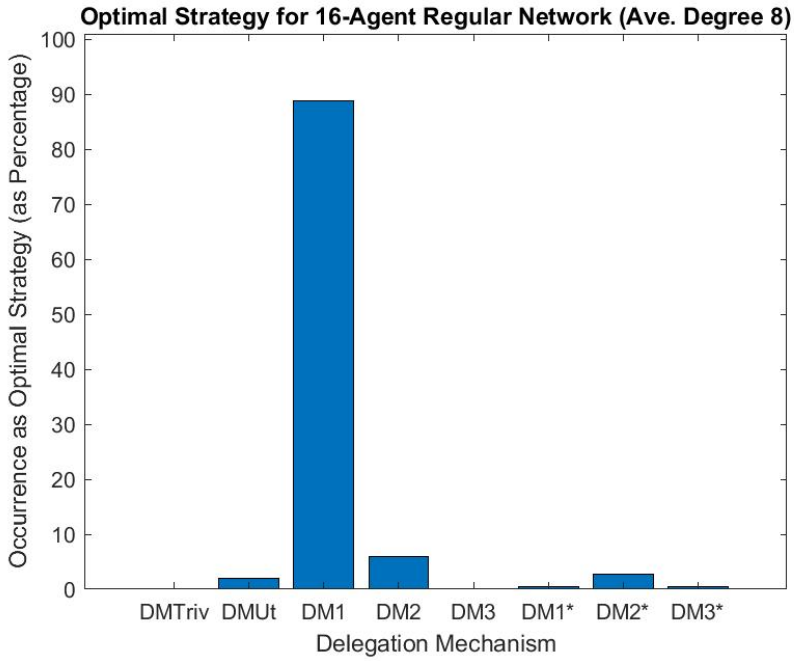

Optimal Strategy for 16-Agent Random Network (Ave. Degree 4)
Occurrence as Optimal Strategy (as Percentage)
0
10
20
30
40
50
60
70
80
90
100
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

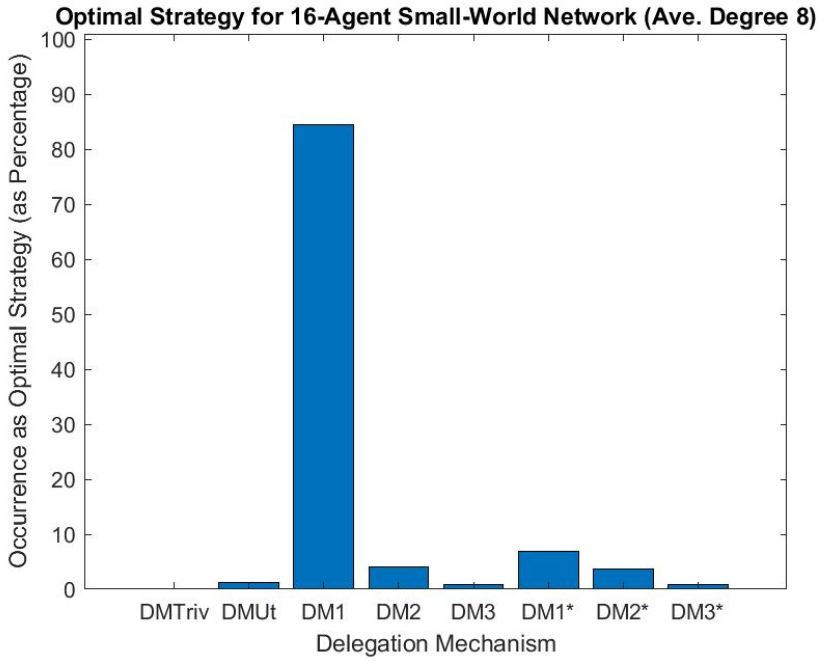

Optimal Strategy for 16-Agent Regular Network (Ave. Degree 4)
Occurrence as Optimal Strategy (as Percentage)
100
90
80
70
60
50
40
30
20
10
0
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

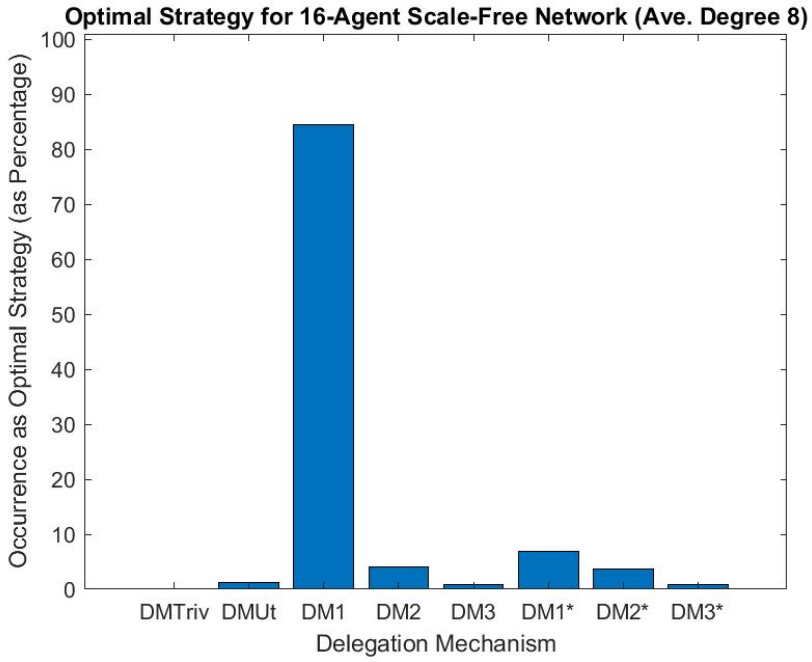

Optimal Strategy for 16-Agent Small-World Network (Ave. Degree 4)
Occurrence as Optimal Strategy (as Percentage)
100
90
80
70
60
50
40
30
20
10
0
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

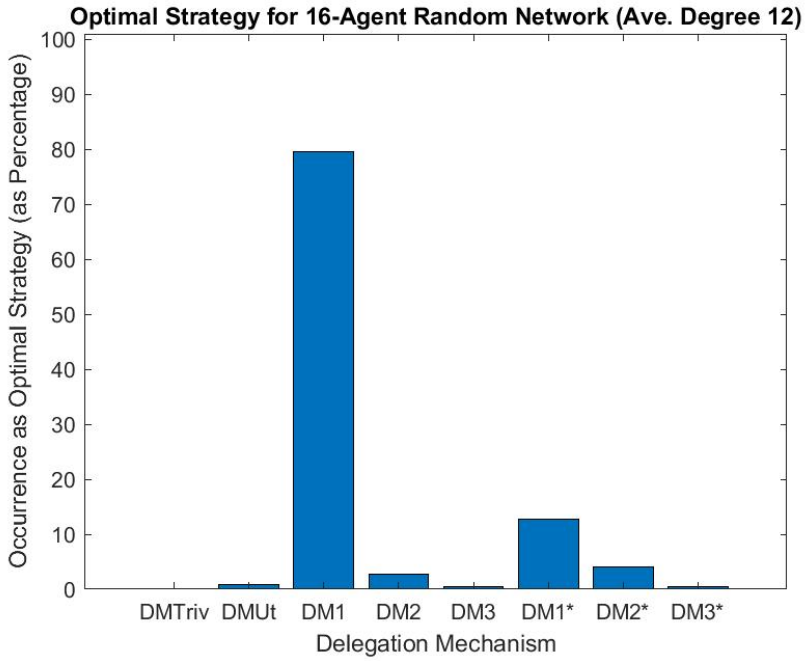

Optimal Strategy for 16-Agent Scale-Free Network (Ave. Degree 4)
Occurrence as Optimal Strategy (as Percentage)
100
90
80
70
60
50
40
30
20
10
0
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

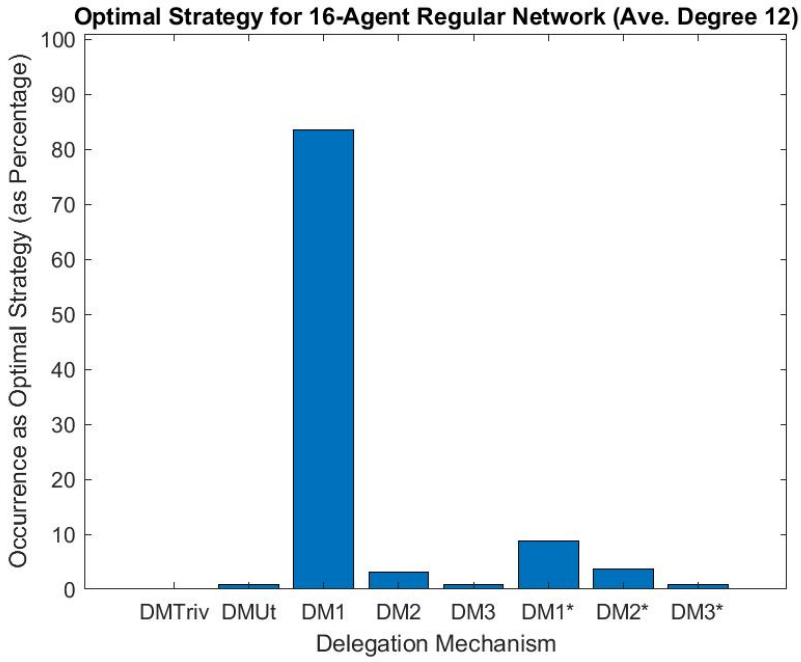

Optimal Strategy for 16-Agent Random Network (Ave. Degree 8)
Occurrence as Optimal Strategy (as Percentage)
100
90
80
70
60
50
40
30
20
10
0
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

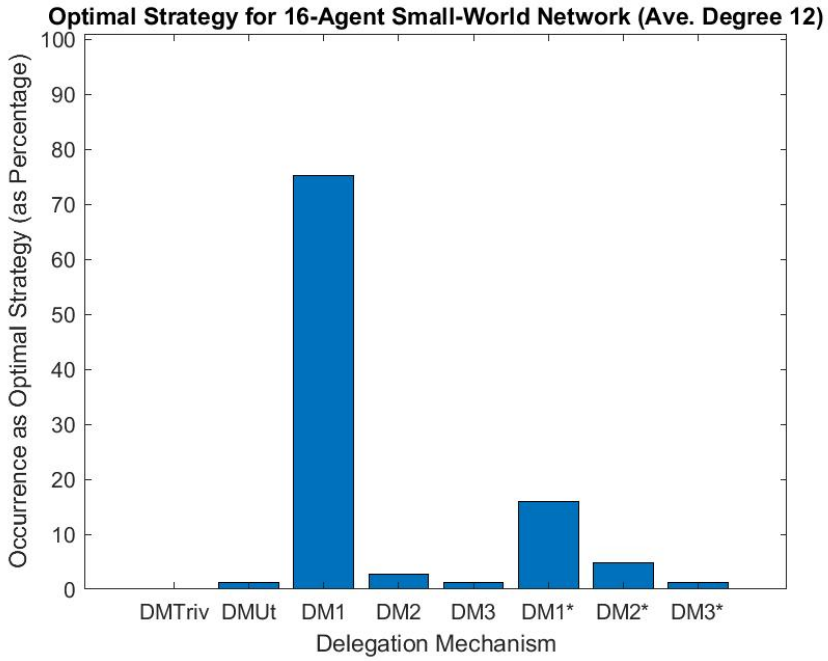

Optimal Strategy for 16-Agent Regular Network (Ave. Degree 8)
Occurrence as Optimal Strategy (as Percentage)
0
10
20
30
40
50
60
70
80
90
100
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

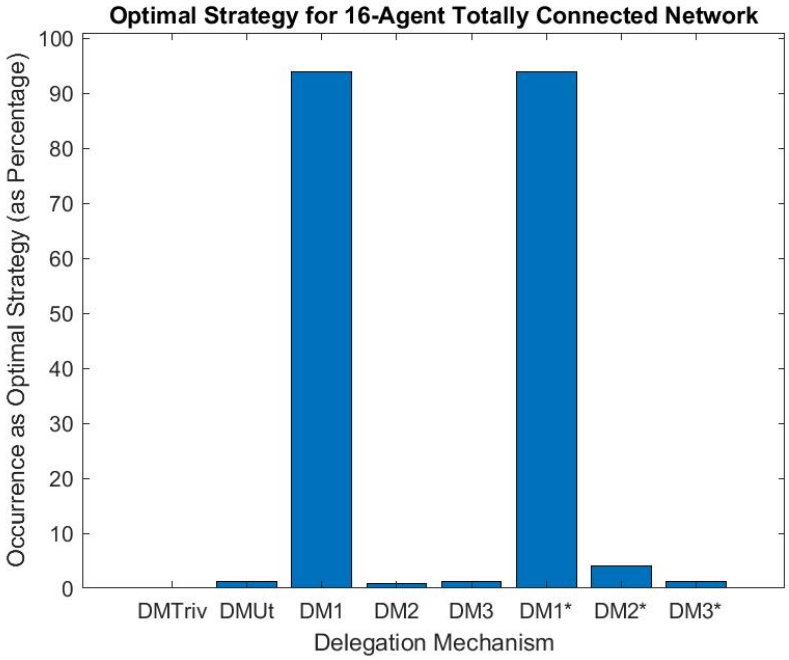

Optimal Strategy for 16-Agent Small-World Network (Ave. Degree 8)
Occurrence as Optimal Strategy (as Percentage)
0
10
20
30
40
50
60
70
80
90
100
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism


Optimal Strategy for 16-Agent Scale-Free Network (Ave. Degree 8)
Occurrence as Optimal Strategy (as Percentage)
100
90
80
70
60
50
40
30
20
10
0
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism


Optimal Strategy for 16-Agent Random Network (Ave. Degree 12)
Occurrence as Optimal Strategy (as Percentage)
100
90
80
70
60
50
40
30
20
10
0
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

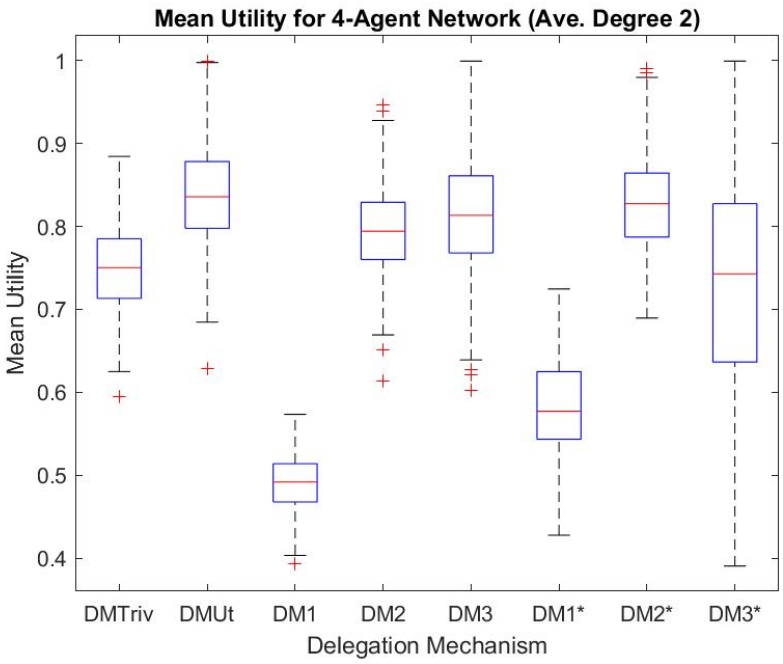

Optimal Strategy for 16-Agent Regular Network (Ave. Degree 12)
Occurrence as Optimal Strategy (as Percentage)
100
90
80
70
60
50
40
30
20
10
0
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

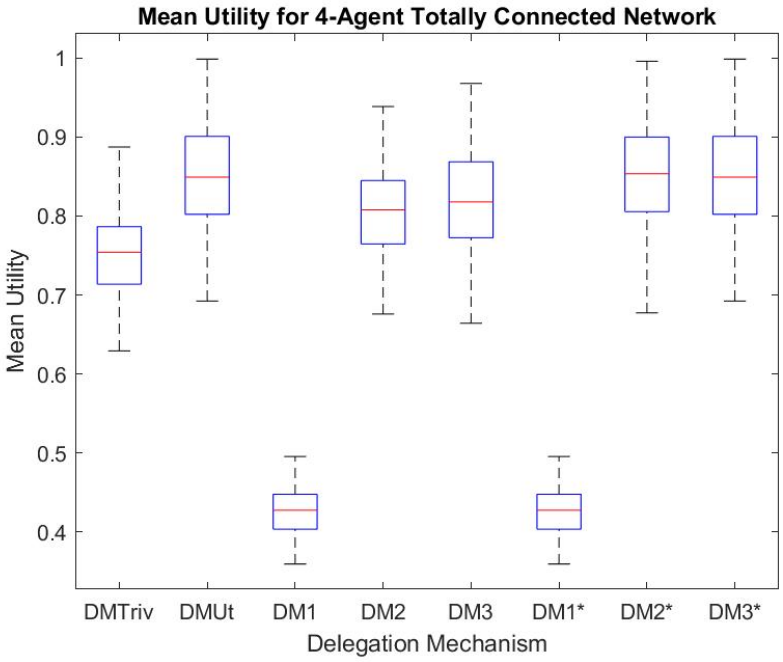

Optimal Strategy for 16-Agent Small-World Network (Ave. Degree 12)
Occurrence as Optimal Strategy (as Percentage)
100
90
80
70
60
50
40
30
20
10
0
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

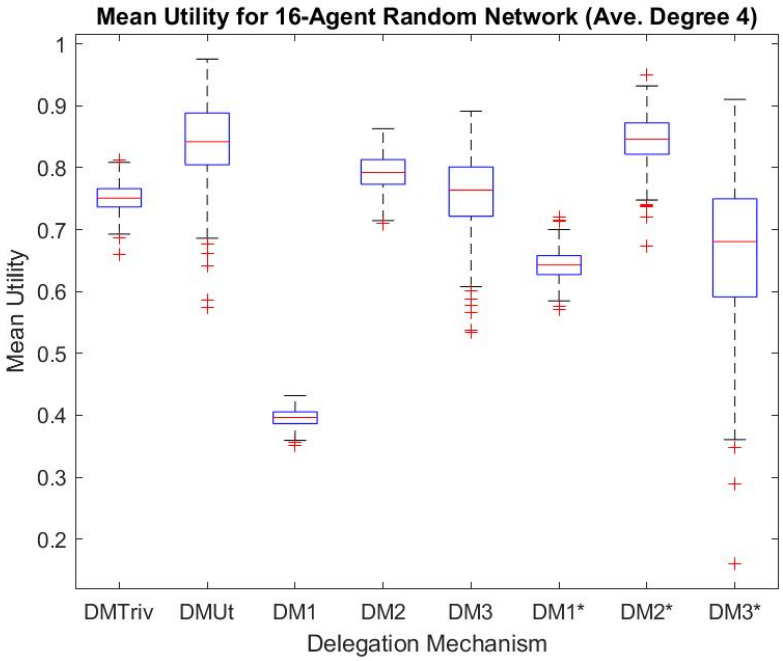

Optimal Strategy for 16-Agent Totally Connected Network
Occurrence as Optimal Strategy (as Percentage)
100
90
80
70
60
50
40
30
20
10
0
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

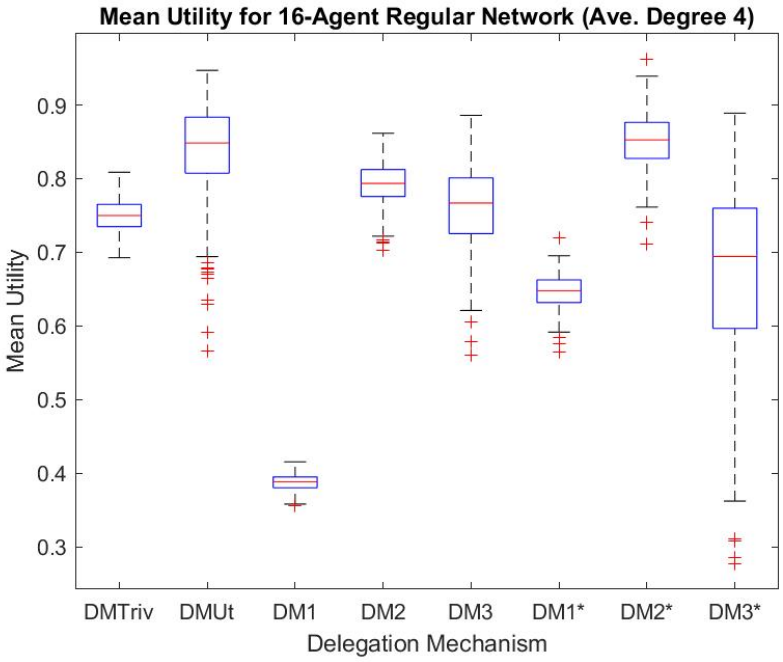
Appendix R

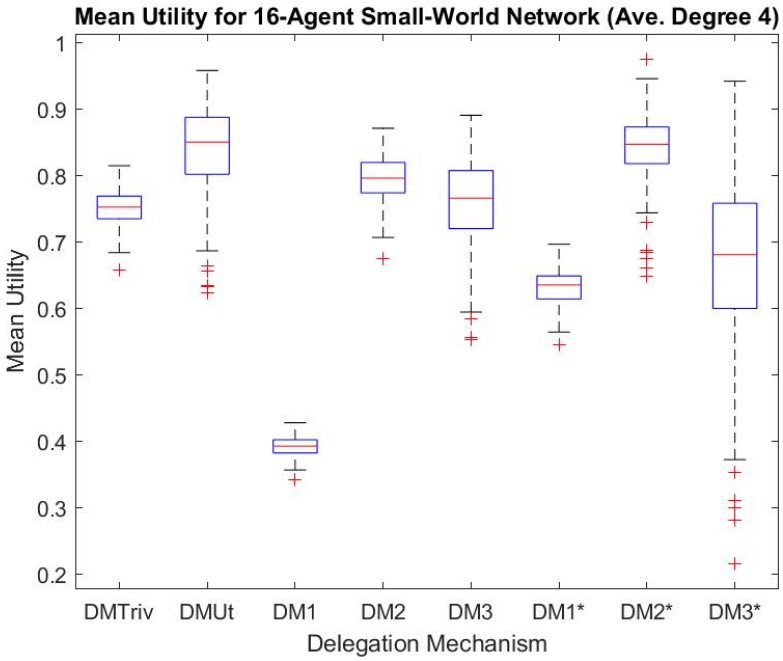
Utility Box Plots

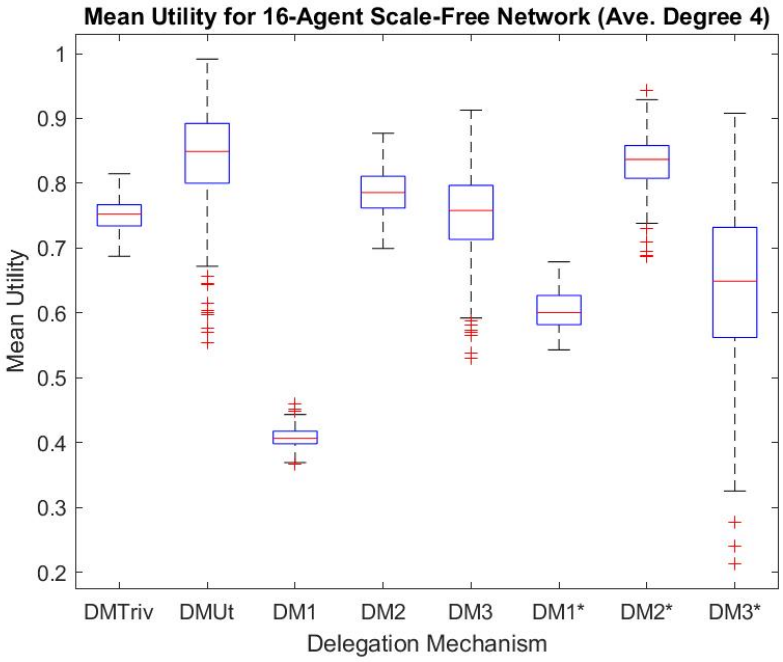


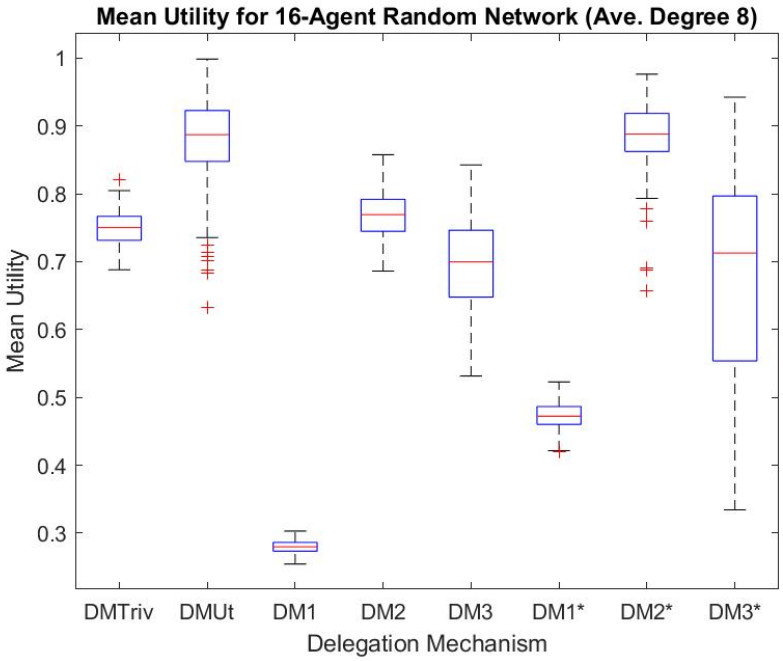

Mean Utility for 4-Agent Totally Connected Network
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

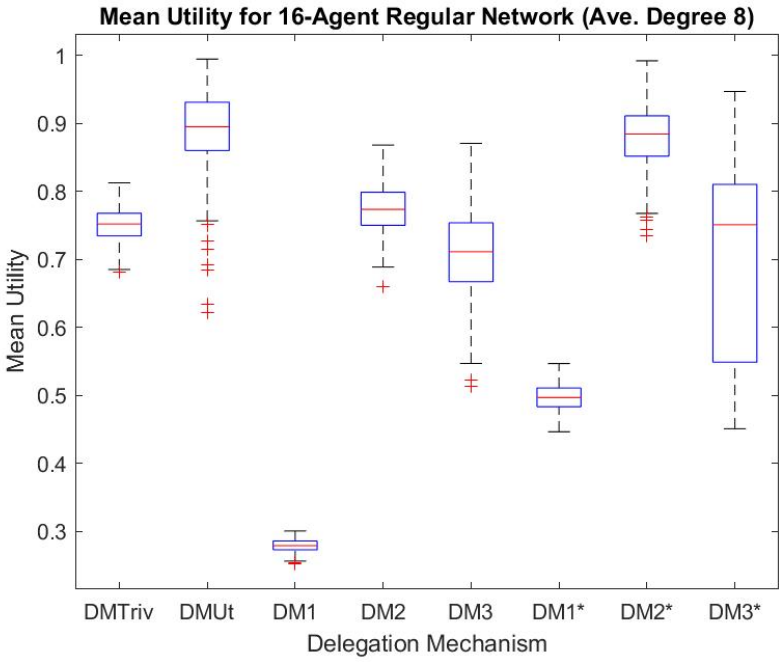

Mean Utility for 16-Agent Random Network (Ave. Degree 4)
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

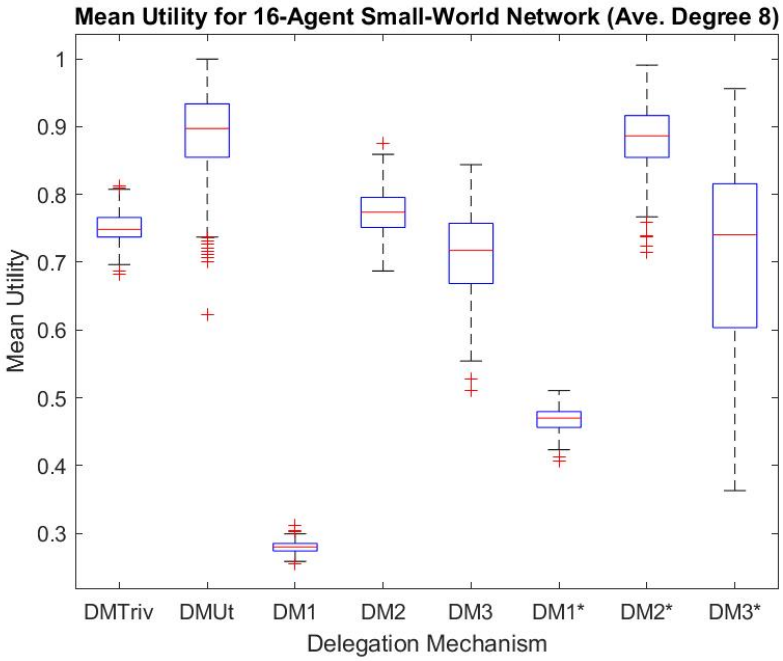

Mean Utility for 16-Agent Regular Network (Ave. Degree 4)
Mean Utility
0.9
0.8
0.7
0.6
0.5
0.4
0.3
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

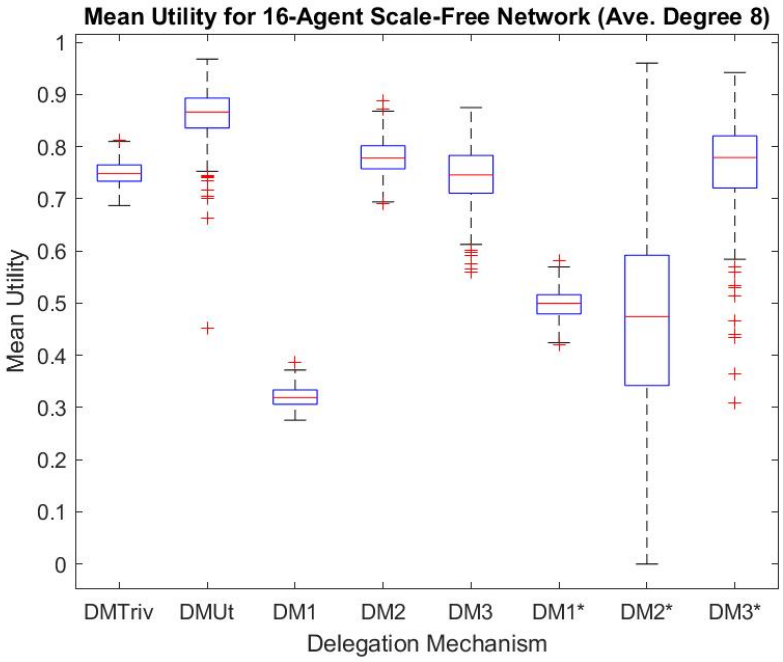

Mean Utility for 16-Agent Small-World Network (Ave. Degree 4)
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

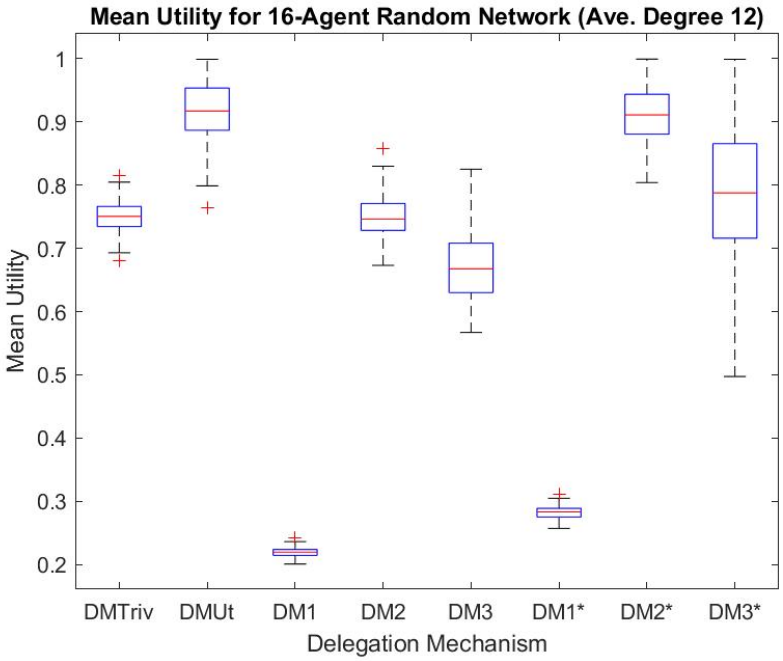

Mean Utility for 16-Agent Scale-Free Network (Ave. Degree 4)
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

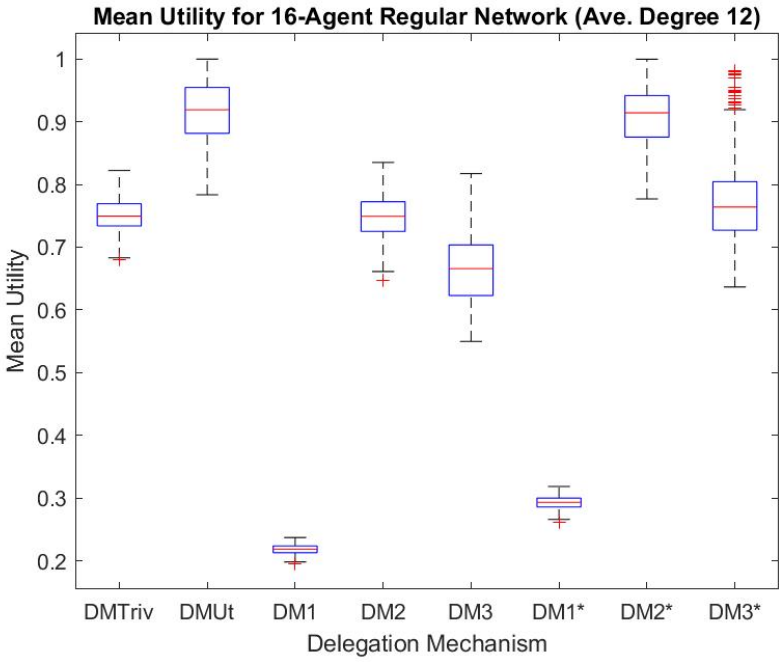

Mean Utility for 16-Agent Random Network (Ave. Degree 8)
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

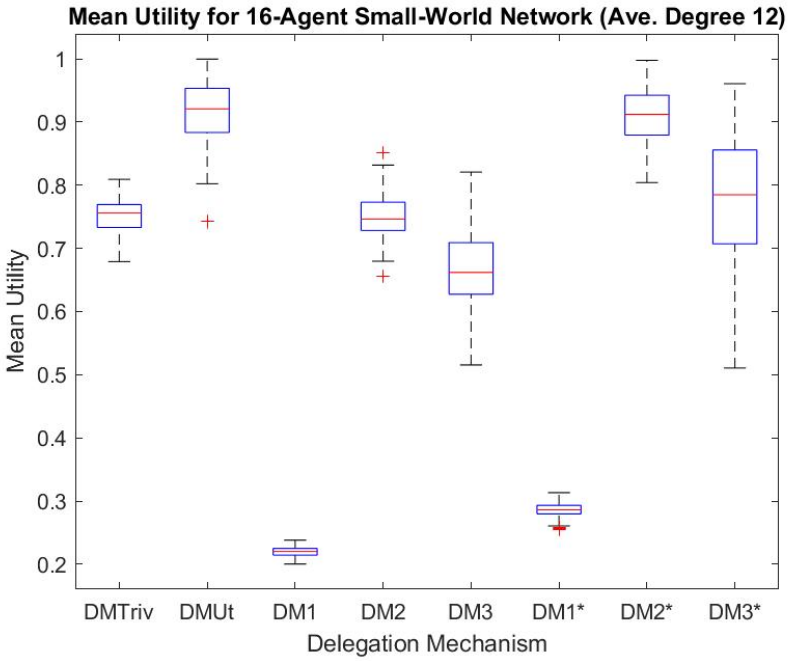

Mean Utility for 16-Agent Regular Network (Ave. Degree 8)
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism

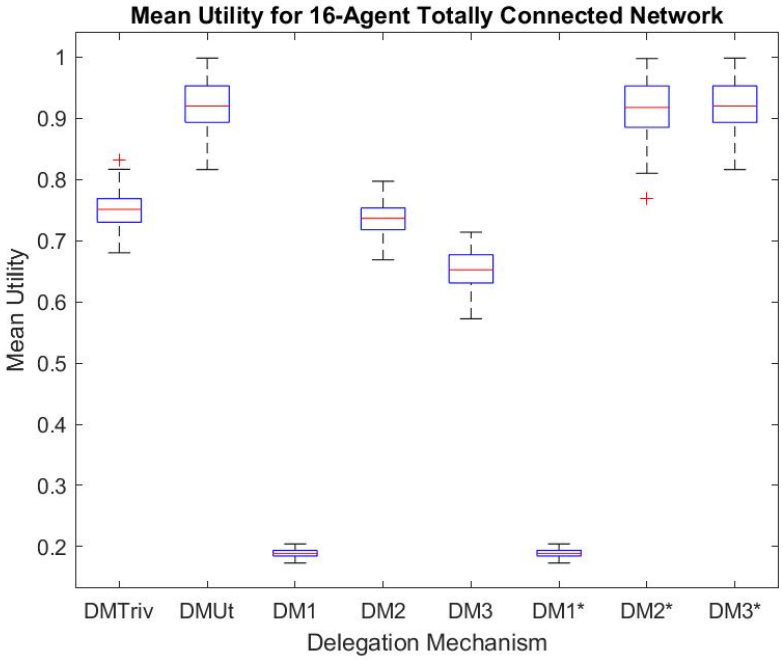

Mean Utility for 16-Agent Small-World Network (Ave. Degree 8)
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism


Mean Utility for 16-Agent Scale-Free Network (Ave. Degree 8)
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism


Mean Utility for 16-Agent Random Network (Ave. Degree 12)
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism


Mean Utility for 16-Agent Regular Network (Ave. Degree 12)
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism


Mean Utility for 16-Agent Small-World Network (Ave. Degree 12)
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism


Mean Utility for 16-Agent Totally Connected Network
Mean Utility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
DMTriv
DMUt
DM1
DM2
DM3
DM1*
DM2*
DM3*
Delegation Mechanism