TURING TEST VIA EMERGENT COMMUNICATION IN THE GAME OF WEREWOLF

Anonymous authors
Paper under double-blind review

ABSTRACT

Emergent communication can lead to more efficient problem-solving heuristics and more domain specificity than a handcrafted communication protocol, potentially directing autonomous agents towards unforeseen yet effective solutions. Previous research has investigated a social deduction game, called Werewolf, where two groups of autonomous agents, villagers and werewolves, interact in an environment named RLupus. We extend it to allow the agents to communicate through multiple rounds and evaluate their language and performance against the baseline environment. We show that agents develop a highly successful heuristic using a single word vocabulary. They create an approach analogous to the Turing test allowing them to determine which agents are werewolves, which is the winning condition. Our experimental analysis shows that our approach speeds up the convergence of the agents towards effective communication strategies.

1 INTRODUCTION

Emergent communication allows the agents to develop their own language as they interact. This allows a bespoke vocabulary/syntax for a given environment and set of actions, allowing for efficient communication and which drives the agents' behaviour. The ability for the language to be completely adapted to the environment also allows for unique and better strategies, as the whole meaning behind the communication channel can be adapted.

Different aspects of how the communication emerges and its properties have been investigated, with one of the first works being that of Lazaridou et al. (2017), where the authors were able to induce language emergence through referential games. Most of the emergent communication literature focuses on relatively simple settings, with the most common being referential games, where two agents have to describe and identify a picture from the description (Lazaridou et al., 2017; 2018; Luna et al., 2020; Dessì et al., 2021). In such settings, there have also been some attempts (Evtimova et al., 2018; Harding Graesser et al., 2019; Kottur et al., 2017) to consider multiround communication, with promising results in terms of the performance and generalisation of the agents' language. However, the work of Kottur et al. (2017) shows that while the language may be effective at solving the tasks that the agents are presented with, it is not necessarily interpretable (i.e., understandable by humans) nor compositional (i.e., consisting of symbols that can be combined to either create a new meaning or convey already learned information).

Emergent communication has been used to improve the performance of intelligent agents in the game of Werewolf (Brandizzi et al., 2021), where players—divided into two teams, werewolves and villagers—have to use their social and deduction skills to identify players from the opposing team. The authors show that the performance significantly improves when the agents create their own language, however the analysis of the language and strategy of the villagers is left for future work. In addition, in the original environment (Brandizzi et al., 2021), the villagers only had a single round of communication to agree on a player to vote out as an alleged werewolf, which may not allow them to fully explore other, possibly better, strategies.

In this paper, we extend the Werewolf environment by introducing multiple rounds of communication. This is to determine the effect of a longer period of discussion on the language developed and the success rate in identifying the werewolf. Our work is motivated by the assumption that if agents are allowed more time to communicate, then they will develop better, more successful strategies, compared to handcrafted solutions.

2 GAME OF WEREWOLF

2.1 WEREWOLF

We use the game of Werewolf as the basis for our work. It begins with the assignment of roles to the players. Some players are assigned the role of werewolves, while others are villagers. The ratio of werewolves to villagers is usually at most 1:3.

The game has two phases in which different actions can be taken—"daytime" and "nighttime". The game begins at "nighttime", and the werewolves are allowed to discuss and choose a villager player to vote out. The discussion and the votes of the werewolves are only visible to the werewolf agents. After a player has been chosen by the werewolves, he is removed from the game, and the game progresses to the "daytime" phase. During this "daytime" phase, all players are allowed to discuss the happenings of the previous phases, including the "nighttime" killing. Then, all players may vote to eliminate a player that they think is a werewolf. After the "execution", the game loops back to the "nighttime" phase, and the phases alternate until either team wins the game.

The goal of the werewolves is to eliminate all villagers, while the goal of the villagers is to work out who the werewolves are and vote them out during the "daytime" phase. The game ends when all werewolves are voted out, or when the number of villagers and werewolves are equal. This condition is introduced because when the numbers are equal on each team, the villagers cannot win any more, as all votes would be at most ties.

We provide a visual representation of the game phases in Appendix A.2, Figure 4. For further details, including a theoretical analysis of the win rates when played with and without emergent communication, we refer to the detailed analysis by Brandizzi et al. (2021).

2.2 AGENT ENVIRONMENT

For our work, we will be basing our reinforcement learning environment on that of Brandizzi et al. (2021). This environment uses the Ray (Moritz et al., 2018) RLLib (Liang et al., 2018) library to train all the villager agents with an asynchronous variant of Proximal Policy Optimization (Schulman et al., 2017), as provided with RLLib (Liang et al., 2018). Our agents are Ray LSTM wrapped (Liang et al., 2018) parametric agents with linear layers. The input for the agents is the observation from the environment itself, written with OpenAI Gym (Brockman et al., 2016), and the output is the action to take in the environment, in our case the message and vote of a given agent. Our main contribution to the Werewolf environment is extending it to a multi-round communication domain. We introduce changes to how the voting works to incentivise quicker convergence on communication between the villager agents.

The multi-round approach is implemented during the "daytime" phase when all agents, including the werewolves, are permitted to communicate. This is done as the werewolves in the original environment have a static policy that they have to use (Brandizzi et al., 2021), and so any communication during the "nighttime" phase would be a waste of computation time. Furthermore, our multi-round communication can be parameterised with the number of communication rounds that the agents are allowed to have.

Secondly, we also incentivise our agents to decide as unanimously and as quickly as possible on the target of their vote. This is done through the agreement loss, as presented by Brandizzi et al. (2021), as well as an indirect loss on the number of rounds taken to reach a conclusion. This loss is implemented through a required threshold for a vote to be valid. If less than X% of agents agree on a target, then the vote is considered invalid, and no players are eliminated. This modification can be viewed as adding an independent judge to count all agent votes, where only a significant majority is permitted to decide whether to vote out an agent or not.

3 EVALUATION OF LANGUAGE & PARAMETERS

Differently from (Brandizzi et al., 2021), our modified version of the environment exposes two additional parameters to explore: the number of rounds and the threshold required for the vote to

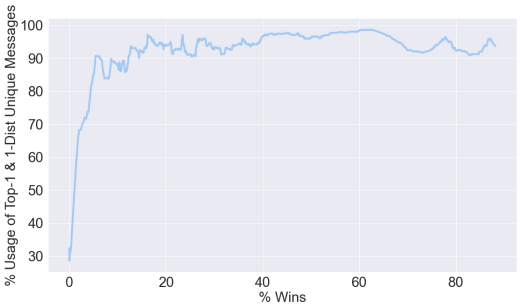

Figure 1: Usage of most used unique message and its Top-10 adjacents vs number of wins. Including the distance one adjacents smooths out most inconsistencies, and shows that our agents may be exploring other strategies by making minor changes to their most common unique message.

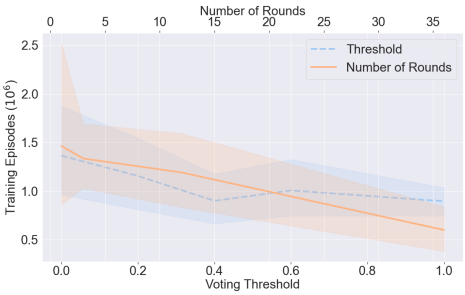

Figure 2: Impact of both the voting majority threshold and number of rounds on the convergence speed. As both parameters increase, we can observe the decrease in the average number of episodes required to reach the 75% win rate. Shaded areas represent the 95% confidence interval.

be valid. We explore them using a grid search, maintaining the other parameters from the original paper (Brandizzi et al., 2021). Further technical details are available in Appendix B.1

Our agent and game configuration show improvements over both the theoretical baseline win rates and the previously demonstrated performance (Brandizzi et al., 2021). We define the win rate as the percentage of games that the villagers win in a single training run. Therefore, if one of our training runs had 100 games, where each game would consist of multiple rotations between "daytime" and "nighttime" phases until one team wins, and the villagers won 89 games, our reported win rate would be 89%.

As shown by Brandizzi et al. (2021), their agents have already improved upon their calculated theoretical baseline win rates. We further improved on these win rates shown in Table 1. We can see that for certain configurations of the round count and threshold, our agents achieve a 100% win rate.

3.1 Turing Test Strategy

We hypothesise we have observed a novel strategy in our villager agents, which is that of a Turing test (Turing, 1950) analogue. Here, the villagers send the same signal every round of communication, and see who does not comply with this strategy. As werewolves do not learn because they have a static policy, this strategy allows villagers to easily distinguish friend from foe. This allows them to easily vote out one of the werewolves in the next round. For this strategy, multiple rounds of communication also help the agents develop this test faster, and to not require a blind vote at the beginning, as they can establish this distinction without having to vote.

Furthermore, we can analyse the language that our agents have developed for this strategy. The languages that have been created seem to be mainly focused on a sparse word vocabulary for any successful strategy. Most unsuccessful strategies do not use a single word, but rather have multiple words in their vocabulary. Moreover, almost all successful strategies use their single word most of the time, with a minor number of outliers. Therefore, our agents do not develop a compositional language, but rather a code.

We can see the usage of the most common word, and its distance one adjacents in Figure 1. The distance one adjacents are defined as any message that has a single character difference from the most common message. We gather this information from the later parts of the training of our agents, as to focus on an already developed strategy, rather than the exploration phase.

3.2 Convergence Speed

We have also observed that our modifications impact the convergence speed significantly. We define the convergence point as the episode number where our agents reach over 75% win rate, which we chose as an arbitrary threshold for a successful strategy. Both the number of rounds and the voting threshold decrease the average convergence episode. We can observe that the higher the number of

Table 1: Results for both our and the original (Brandizzi et al., 2021) environment. Our modifications achieve a lower convergence point and a higher win rate for 2/3 configurations. The better values are displayed in **bold**. Convergence episode for the original configurations is reported as obtained by our reproduction.

Configuration	Threshold	Rounds	Our WR	Original WR	Convergence
SL9-SR2-PL9	1	36	**100%**	N/A	**706962**
SL9-SR2-PL9	0	1	99%	45%	1296800
SL9-SR2-PL21	1	3	96%	N/A	**704879**
SL9-SR2-PL21	0	1	95%	**98%**	1199808
SL21-SR2-PL21	0.4	3	**100%**	N/A	625122
SL21-SR2-PL21	0	1	98%	94%	**587084**

communication rounds, the quicker the agents converge. Furthermore, we can see that the average convergence speed increases as the voting threshold increases. We can see both of these relationships in Figure 2. This means that enforcing a higher agreement rate between agents could also positively impact the speed with which they develop a common language or code. Furthermore, we can note from Figure 2, that the number of rounds positively affects convergence and decreases the amount of time that is required to find a good strategy.

We also performed statistical analysis of the convergence speed. Our results show that the number of rounds has a statistically significant effect on both win rate and convergence speed. This further confirms that the more time the agents have to converse, the quicker they can converge on a common shared strategy. However, the voting threshold may not have a statistically significant effect. For the details of our analysis, please see Appendix C.

3.3 Comparison to the Original Environment

The authors of RLupus (Brandizzi et al., 2021) present interesting findings in terms of how emergent communication affects performance in the game of Werewolf. They show that, with certain configurations for the message length and vocabulary size, the agents which are allowed to create their own language will outperform other agents. Further, when compared with the AI Wolf competition (Toriumi et al., 2017), the emergent communication agents surpass their handwritten counterparts.

In Table 1 we compare our results to the original environment. Our configuration notation is consistent with Brandizzi et al. (2021), where "SR" is signal range, "SL" is signal length and "PL" is the number of players. Signal range refers to the number of characters available (i.e., 2 is binary), and signal length refers to the message length. "WR" refers to "Win Rate". Lastly, the "Convergence" column refers to the episode number, or convergence point, that the agents with that particular configuration achieved.

With our modifications to the environment, we achieve a lower total episode count before convergence for two out of three compared configurations, while also achieving a higher win rate for two out of the three configurations. This shows that our modifications in terms of voting threshold and multi-round communication offer to improve both metrics of the game. We have also included reproductions of the configurations that were presented by Brandizzi et al. (2021) with our modifications to the code for a better comparison.

4 Conclusions

With the arrival of deep-learning, emergent communication is an area of research that is currently having a resurgence of interest. The aim is to allow agents to develop their own communication, that is optimised for the setting they operate in. The benefits are efficiency, reduced designer effort and creativity in problem-solving. We have introduced multi-round communication to the originally single round environment of Werewolf, as presented by Brandizzi et al. (2021), to study these properties. We show that the number of communication rounds affect the convergence time of the agents, with statistical analysis to back our claim. Finally, we investigate the strategies that the agents develop to achieve the high win rates we observed, and show that our agents may be exhibiting a behaviour that resembles the Turing test with the multi-round interactions allowing villages to establish this effective identity test faster.

REFERENCES

N. Brandizzi, D. Grossi, and L. Iocchi. RLupus: Cooperation through emergent communication in The Werewolf social deduction game. *Intelligenza Artificiale*, 15(2), 2021.

G. Brockman, V. Cheung, L.g Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs]*, 2016.

R. Dessì, E. Kharitonov, and M. Baroni. Interpretable agent communication from scratch(with a generic visual processor emerging on the side). *arXiv:2106.04258 [cs]*, 2021.

K. Evtimova, A. Drozdov, D. Kiela, and K. Cho. Emergent Communication in a Multi-Modal, Multi-Step Referential Game. In *ICLR*, 2018.

L. Harding Graesser, K. Cho, and D. Kiela. Emergent Linguistic Phenomena in Multi-Agent Communication Games. In *EMNLP*, pp. 3700–3710, 2019.

S. Kottur, J. M. F. Moura, S. Lee, and D. Batra. Natural Language Does Not Emerge 'Naturally' in Multi-Agent Dialog. In *EMNLP*, pp. 2962–2967, 2017.

A. Lazaridou, A. Peysakhovich, and M. Baroni. Multi-Agent Cooperation and the Emergence of (Natural) Language. In *ICLR*, 2017.

A. Lazaridou, K. M. Hermann, K. Tuyls, and S. Clark. Emergence of Linguistic Communication from Referential Games with Symbolic and Pixel Input. In *ICLR*, 2018.

E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica. RLlib: Abstractions for Distributed Reinforcement Learning. In *ICML*, 2018.

D. R. Luna, E. M. Ponti, D. Hupkes, and E. Bruni. Internal and External Pressures on Language Emergence: Least Effort, Object Constancy and Frequency. In *EMNLP*, pp. 4428–4437, 2020.

W. McKinney. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, 2010. doi: 10.25080/Majora-92bf1922-00a.

P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica. Ray: A Distributed Framework for Emerging AI Applications. *arXiv:1712.05889 [cs, stat]*, 2018.

The pandas Development Team. pandas-dev/pandas: Pandas, February 2020. URL https://doi.org/10.5281/zenodo.3509134.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017.

F. Toriumi, H. Osawa, M. Inaba, D. Katagami, K. Shinoda, and H. Matsubara. AI Wolf Contest — Development of Game AI Using Collective Intelligence —. In *Computer Games*, Communications in Computer and Information Science, pp. 101–115, 2017.

A. M. Turing. Computing Machinery and Intelligence. *Mind*, LIX(236), 1950.

P. Virtanen, R. Gommers, T.s E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C J Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 2020.

A APPENDIX: FURTHER ANALYSIS

These results are extremely promising for the language that the agents have developed and the possible tuning of parameters in the future. We can discuss the other observations that we have identified during this work.

A.1 PRESSURES

Firstly, we can look into the pressures that our agents have for winning the game. Regarding the 9-player configuration, where there are 6 villagers and 3 werewolves, the pressure on the villagers to communicate efficiently and accurately is high. This is because a single mistake in terms of the vote on the villager part leads to a loss of the game. To illustrate this, we can consider the following example:

The game begins with a "nighttime" phase, where the werewolves get to vote out a villager. This brings down the number of villagers to 5, with 3 werewolves remaining. Now the game moves to the "daytime" phase, where the villagers may vote. If the villagers vote correctly and remove a werewolf, the game continues with 5 villagers and 2 werewolves. However, if the villagers vote for a villager, the game moves to the "nighttime" phase with 4 villagers and 3 werewolves. At this point, the werewolves vote out one villager, and the game ends with the werewolves winning. A similar situation happens even if the villagers get the first vote correct. This means that the villagers cannot make a single mistake in this configuration.

A.2 WIN RATE

Furthermore, we can look at the direct impact that both of our additional parameters have on the average win rate of the villagers. Figure 3 illustrates the interaction between the parameters in terms of the win rate, with the shaded areas representing the 95% confidence interval.

When looking purely at the effect of the number of rounds, for a larger amount the win rates tend to be lower, as we can see in Figure 5. However, the number of rounds together with the voting threshold increases the average win rate past the previously reported values (Brandizzi et al., 2021), and results in an overall positive trend.

We hypothesise that this decrease is due to the longer training times, as the larger number of rounds requires more computation, and therefore does not achieve convergence within the same time as our other configurations. We can see that the number of total episodes decreases as the number of rounds increases, which is due to the limited amount of time that we could run the simulations in Figure 6.

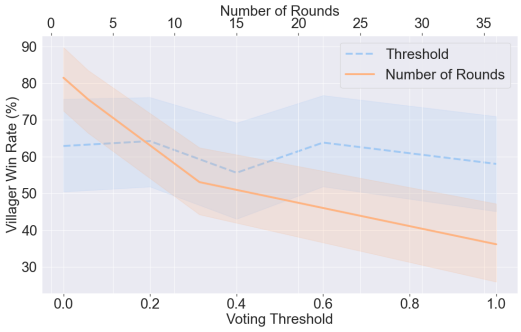

Figure 3: Impact of both the voting majority threshold and number of rounds on the win rate. While the number of rounds decreases the average win rate, the threshold's impact is less negative.

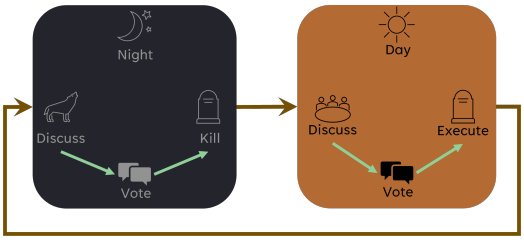

Figure 4: Visual representation of the flow of the game of Werewolf. The game starts at the night phase, progresses to the day phase, and then loops back around to the night phase. The discussion in the night phase only includes werewolves, while in the day phase all players may converse.

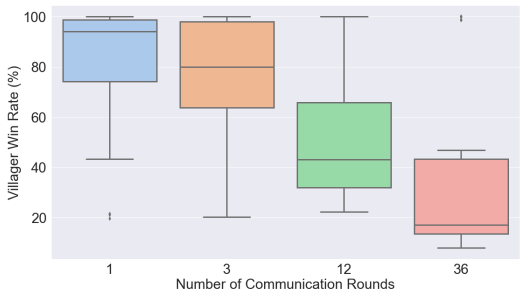

Figure 5: Impact of the number of communication rounds on the win rate of villagers. We can see that the number of communication rounds is negatively correlated with the average win rate.

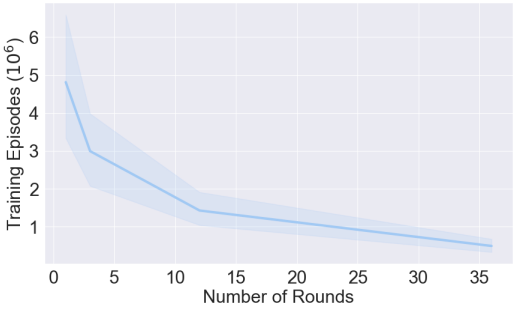

Figure 6: Impact of the number of communication rounds on the number of training episodes. We hypothesise that this is the reason behind lower performance as the number of rounds increases. Due to a bigger simulations size, as the agents need to exchange more messages, the configurations with higher number of rounds perform worse, as they do not have enough time to converge.

B Appendix: Technical Details

B.1 Hyperparameters & Training Details

For our parameter grid, we have used the values of [0, 0.2, 0.4, 0.6, 1] and [1, 3, 12, 36] for the threshold and number of rounds respectively. We run our simulations for an average of 3M episodes. With our experimental setup as described, we have performed a total of 130 runs for a total of 2600 hrs of GPU time. An average run of 3M episodes took 20h on a single NVIDIA RTX8000 GPU.

B.2 Pseudocode

The flow for our environment is as described in Algorithm 1, and visualised in Figure 4. As per the pseudocode, the limit for the number of in-game days is ten, where a day consists of a "nighttime" and "daytime" phases, both split into communication and voting. An episode in our environment is defined as a single play of the game, when either werewolves or villagers win, which can be up to ten in-game days.

To summarise the pseudocode, we begin with setting the initial values for the day and the phase to zero. Then we enter the main loop, where we will be running the multiple "daytime" and "nighttime" phases. This loop will exit when day ten is reached, or when a win condition is reached, as per our description of the game. Inside the loop we run the four distinct phases — Nighttime Communication, Nighttime Voting, Daytime Communication and Daytime Voting. Nighttime communication is for the werewolves to communicate, however as they have a static policy this is mostly a placeholder. Similarly, the Nighttime voting is exclusively for the werewolves, and they use their static policy to choose a target, in our case a random target. The "Voted" function returns the agent with most votes, referred to as "Dead Agent" and the agents who were in the minority, referred to as "Non Accord". The agent who was voted for by the werewolves receives a negative reward, and the game moves on to the "Daytime" phase. Here all agents are allowed to communicate, and then all agent may vote. We use the same "Voted" function to gather the "Dead Agent" and "Non Accord" agents if the vote was successful; otherwise we give a negative reward to all villagers as the vote was invalid. If the vote was successful, the dead agent receives a negative reward, as do the agents who were not in the majority.

The game then progress in this fashion until an exit condition is reached for the loop. After the loop exits, all rewards are distributed, including the win/lose reward.

Algorithm 1 Flow of an Episode of the Werewolf Game

```
Day ← 0
Phase ← 0
while Day ≠ 10 do
    if Phase = NighttimeCommunication then
        Werewolves communicate
        Phase ← Phase + 1
    else if Phase = NighttimeVoting then
        Dead_Agent, Non_Accord ← Voted()
        Rewards[Dead_Agent] ← Rewards[Dead_Agent] − 5
        Phase ← Phase + 1
    else if Phase = DaytimeCommunication then
        All agents communicate
        Phase ← Phase + 1
    else if Phase = DaytimeVoting then
        All agents vote
        if Vote ≥ Threshold then
            Dead_Agent, Non_Accord ← Voted()
            Rewards[Dead_Agent] ← Rewards[Dead_Agent] − 5
            for all V ∉ Non_Accord do
                Rewards[V] ← Rewards[V] − 1
            end for
        else
            Vote is invalid, and no action is taken
            for all V ∈ Villagers do
                Rewards[V] ← Rewards[V] − 2
            end for
        end if
        Phase ← 0
        Day ← Day + 1
    end if
end while
if Agents Won then
    Rewards[All] ← Rewards[All] + 25
else
    Rewards[All] ← Rewards[All] − 25
end if
Distribute Rewards
```

Table 2: Normality Test Results. Our data is **not** normally distributed for both the win rate and the episode count.

Variable	Test Type	p-Value	Statistic
Win Rate	Shapiro-Wilk	0.000	0.858
Win Rate	D'Agostino and Pearson	0.000	2554.462
Episode Count	Shapiro-Wilk	0.000	0.659
Episode Count	D'Agostino and Pearson	0.000	67.593

C APPENDIX: STATISTICAL SIGNIFICANCE

To verify our results, we have performed a statistical significance analysis. This analysis relies on the 130 data points that we have obtained through our simulations. However, as the number of data points per configuration are relatively low, the overall results for the regression analysis may not be fully accurate.

Firstly, we analysed the normality of the distribution of both of our dependent variables - the mean of villager win rates, and the number of episodes that it takes villagers to converge. We performed the normality analysis with two tests from the SciPy (Virtanen et al., 2020) package. We report the results of the tests in Table 2.

Secondly, we analysed the correlation and significance of the correlation with the Spearman correlation metric, as our data was not normally distributed. We used the function provided by the pandas (McKinney, 2010; pandas Development Team, 2020) package. We can see the results of these tests in Figure 7.

Finally, we analysed our results with a simple linear regression model, from the SciPy (Virtanen et al., 2020) package. We present the results of this analysis in Table 3.

Overall, from our analyses we can infer that the number of rounds does have a statistically significant effect on both the win rate and convergence speed, as we have theorised in the main body of this work. However, our analysis fails to find a relationship between the voting threshold and either of our dependent variables.

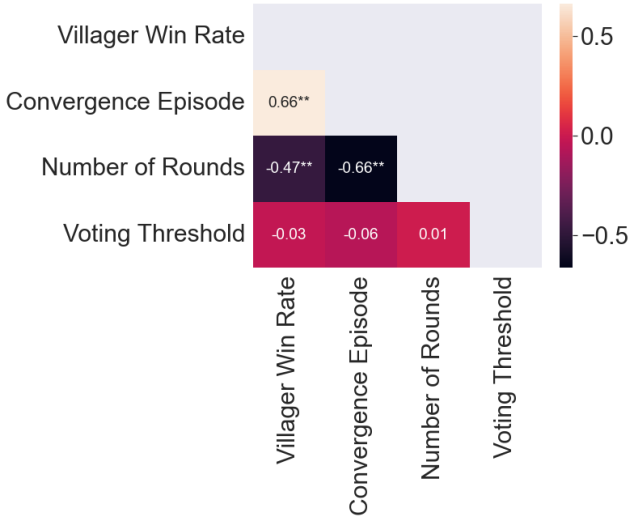

Figure 7: Spearman correlation strength and its significance. The strength for the number of rounds affecting either the villager win rate or convergence episode is high. However, the relationship between the threshold and win rate or convergence is much weaker. The significance can be discerned by the number of * next to the corresponding number, where **no** * signifies $p > 0.05$; * is $p < 0.05$; and ** is $p < 0.01$

Table 3: Linear Regression Analysis. The win rate and convergence episode are very likely to be affected by the number of rounds. However, similarly to the previous tests, the voting threshold may not have an effect on either.

Relationship	p-Value	R^2
Number of Rounds vs Win Rate	< 0.001	0.264
Voting Threshold vs Win Rate	0.600	0.002
Number of Rounds vs Convergence Episode	< 0.001	0.189
Voting Threshold vs Convergence Episode	0.686	0.001