Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

ON THE MODELING AND AGENT-BASED SIMULATION OF A COOPERATIVE GROUP ANAGRAM GAME

Zhihao Hu, Xinwei Deng
Brian J. Goode, Naren Ramakrishnan
Parang Saraf, Nathan Self

Virginia Tech
Blacksburg, VA 24061, USA

Abhijin Adiga, Gizem Korkmaz
Chris J. Kuhlman, Dustin Machi
Madhav V. Marathe, S. S. Ravi

University of Virginia
Charlottesville, VA 24061, USA

Yihui Ren

Brookhaven National Laboratory
Upton, NY 11973, USA

Vanessa Cedeno-Mieles

Virginia Tech
Escuela Superior Politécnica del Litoral, ESPOL
Guayaquil, ECUADOR

Saliya Ekanayake

Lawrence Berkeley National Laboratory
Berkeley, CA 94720, USA

ABSTRACT

Anagram games (i.e., word construction games in which players use letters to form words) have been researched for some 60 years. Games with individual players are the subject of over 20 published investigations. Moreover, there are many popular commercial anagram games such as Scrabble. Recently, cooperative team play of anagram games has been studied experimentally. With all of the experimental work and the popularity of such games, it is somewhat surprising that very little modeling of anagram games has been done to predict player behavior/actions in them. We devise a cooperative group anagram game and develop an agent-based modeling and simulation framework to capture player interactions of sharing letters and forming words. Our primary goals are to understand, quantitatively predict, and explain individual and aggregate group behavior, through simulations, to inform the design of a group anagram game experimental platform.

1 INTRODUCTION

1.1 Background and Motivation

Anagram games, or word construction games, consist of players forming words from a provided group of letters. Research on anagram games—*individual* anagram games—has a long history that dates back at least to 1958, and encompasses more than 20 works that study a variety of issues. Moreover, there are many popular anagram games that are typically played by competing *individuals*, such as Scrabble, Bananagram, and Upwords. Recently, Charness et al. (2014) introduced a *group* anagram game (GrAG), where players cooperate to form words. See Section 2 for details.

Considering the substantial use of anagram games, it is surprising that almost no work has been done in *modeling* and *simulating* these games. In particular, we are interested in modeling GrAGs, notably player interactions and inter-dependence, and the implications of these interactions. There are several general reasons to prefer computational modeling over (laboratory) experiments, e.g., the ability of a validated model to perform computational experiments much faster and at lesser cost. Beyond general motivations, there are several reasons that are expressly related to anagram games, including: (i) modeling GrAGs can be a precursor to modeling other phenomena such as team unity (Charness et al. 2014); and (ii) GrAGs have much in common with other situations in which individuals may share resources in order to mutually benefit (e.g., how to cope during crises, such as hurricanes and forest fires, along with others who remain behind (Yang et al. 2019)). Finally, *a primary motivation for our modeling and simulation work is to predict and understand individual and group performance (e.g., aggregate temporal changes in numbers of words formed, letters requested, and letter replies) in this game in order to provide insights for designing a GrAG software platform for conducting GrAG experiments.*

1.2 Our Group Anagram Game (GrAG)

An overview of the GrAG is given here; details are provided in Section 3. The GrAG is a game played among several players that work cooperatively to form words. They share letters with their immediate neighbors (players are arranged in a network) who use them to form more words than they could form using only their own allotment of letters. Figure 1 shows an illustrative conceptual view of the pair-wise interactions among four players over three time steps. The stated goal of the game is for the team to form as many words as possible. This is because the team's earnings in the game are directly proportional to the number of words that the team forms in total. All players split the earnings evenly, regardless of their performance in the game (e.g., regardless of how many words a particular player forms). This is done to motivate the players to cooperate. Hence, forming the greatest number of words is equivalent to players trying to maximize their earnings.

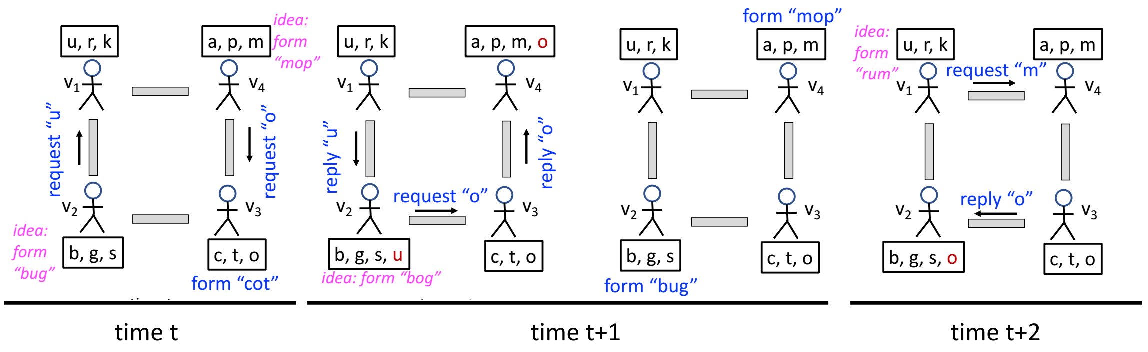


Figure 1: Illustration of the group anagram game (GrAG) setup, and interactions among the four networked players over three time steps. Gray thick lines represent communication channels over which players can request letters from their distance-1 neighbors, and receive them if/when they are sent. Each player v_i has a box containing a sequence L_i^{ih} of letters in-hand (i.e., letters that they can use) to form words. Owned letters (i.e., those initially assigned to a player) are in black and received letters from neighbors are in brown. For example, at time $(t+1)$, v_2 has $L_i^{ih} = (b,g,s,u)$; u is a received letter. Player actions (in blue) are form words, request letters from neighbors, reply to neighbor letter requests, and think (think not shown; it is a no-op). Player motivations are in magenta, e.g., at time t, v_2 gets the idea to form *bug*, and so requests u. A letter received from another player vanishes once it is used (see Section 3 for details).

We emphasize that our work is not modeling cognitive processes within agents that determine agent actions. Rather, we model differences among players' actions by changing their probabilities of taking actions. As will be explained later, greater probabilities result in more actions by players.

1.3 Novelty of Our Work

The novelty of our work is in modeling our GrAG. There is essentially no work in modeling (and thereby predicting) individual behavior in individual anagram games, and there is only one work (Ren et al. (2018)) on predicting player actions in a GrAG. Our GrAG specification differs in an important way from Ren et al. (2018) because in our setup, a player must request a letter for each use, which leads to far more interactions among players. Also, our modeling techniques and the quantities that we predict are also markedly different from this previous work; in our models, a player's actions are based on particular words being formed and the particular letters a player needs (from her neighbors) to form these words (see Section 2). We also propose and evaluate performance measures for game players in our simulations, which has not been done for a GrAG. Even though our model is simplified in some respects (in the absence of data to support more sophisticated models), it gives rise to interesting individual and group dynamics.

1.4 Our Contributions

1. Model of a group anagram game (GrAG). A model for the GrAG has been developed. The model is presented as a set of algorithms in Section 4, useful for understanding the model and for presenting the agent-based simulation (ABS) formulation. The model has different types of parameters that can be tuned to study different behaviors. These are: (*i*) word corpus for forming valid words; (*ii*) the number of initial letters assigned to agents (players) and the particular letter assignments; (*iii*) the network of players (the number of players and the communication channels); and (*iv*) propensities for players to take particular actions (e.g., form word, request a letter, reply to a letter request).

2. Simulations and results for various parameters. We construct software modules of the game models and exercise them in an high performance computing (HPC) agent-based modeling and simulation (ABMS) framework to understand game player performance (Section 5). We evaluate parameters such as network density and number of letters assigned to each player. Several interesting phenomena arise. For example, *reducing* the number of communication channels for a player, from ten to two, *increases* the number of words generated by the team, for many simulation conditions, because players do not get overloaded with requests that they cannot respond to (in a timely manner).

3. Player performance evaluation. We devise and compute simple performance metrics for agents in the game (Section 4). We study factors that lead to changes in player performance, including activity level of players and numbers of letters assigned to players (Section 5).

4. Use of modeling and simulation results. Modeling results are used in two ways in Section 6. First, the computational results provide insights into parameters and parameter values that help guide specification of game conditions for an online game platform. Second, the computational findings enable hypotheses to be formed that can then be tested (with the aforementioned game platform).

Paper organization. Related work is in Section 2. A detailed description of the experimental setup appears in Section 3. The models of the GrAG and player performance are provided in Section 4. Simulation results of modeling the game are in Section 5. Uses of the modeling results are described in Section 6. A summary with limitations of our study concludes the work.

2 RELATED WORK

Individual anagram games. Research on *individual*-based anagram games has a long history, with over 20 research studies. Mayzner and Tresselt (1958) conducted experiments where players had to unscramble letters to form a unique word. Their goal was to evaluate the degree of scramble of letters, quantified by an edit distance, and its effect on time to form a word. Individual anagram games, where a player forms words with provided letters within a specified time, have been used to study performance anxiety (Russell and Sarason 1965), goal achievement and attributing success or failure (Feather and Simon 1971), and whether people prefer pay that is or is not tied to performance (Cadsby et al. 2007).

Group anagram games. Face-to-face *group* anagram games were performed recently in Charness et al. (2014). That work is purely experimental, where players sit at a table and cooperatively form words with a fixed collection of letters. The GrAG is used as a priming activity to foster a sense of unity within the group. The one modeling work on GrAGs is Ren et al. (2018). They develop a time-sequence model that predicts the *types* of actions a player forms as the game progresses. Their predictions are of the type "player v_i selects action type 'form word' at time t." The action is assumed to take place, without regard for the letters that v_i or v_i's neighbors possess. For example, if a player has letters g, b, and x, and the predicted action is form word, then their model will still form an unspecified word from these letters. Our model accounts for letters that players and their neighbors possess and only executes actions if it is possible to do so. Moreover, our GrAG has a key difference. In the game of Ren et al. (2018), neighboring letters only have to be requested once, because each possessed letter is assumed to be in infinite supply. That is, once a player v_i receives the requested letter a, for example, v_i can use a in any number of words; the letter is never exhausted. In our game, however, each use of a neighbor's received letter is consumed so that letters must be requested and received for each use. This leads to many more player interactions.

3 GROUP ANAGRAM GAME (GrAG) DESCRIPTION

Game configuration. A number n of players and a fixed graph $G(V,E)$ on these players is specified, where $v_i \in V$, $i \in \{1,2,\ldots,n\}$, $n = |V|$. The edge set E represents a set of undirected channels such that edge $e = \{v_i, v_j\}$ means that players (nodes) v_i and v_j can communicate. Each player is initially assigned n_ℓ alphabetic letters, either at random or deliberately, and although each player can have a different number of letters, for exposition, it is assumed that all players receive the same number of (initial) letters. See Figure 1.

Player actions in game. As time marches forward, every player can take the actions identified in Table 1 any number of times and in any order within the game duration t_g. The set A of actions is $A = \{a_1, a_2, a_3, a_4\}$, and includes requesting letters, replying to letter requests, and forming words.

Table 1: Actions $a_i \in A$ of players $v_i, v_j \in V$ in the GrAG. Note that each of actions a_2 *request letter* and a_3 *reply with letter* involves two effects on players. A letter request is *sent* by one agent and *received* by another. A letter reply is *sent* by one agent and *received* by another. These ideas of *send* and *receive* are prominent in the models and simulations, as are the actions.

Item	Variable	Action Name	Description
1	a_1	form word	v_i forms and submits a word.
2	a_2	request letter	v_i requests a letter ℓ from a neighbor v_j.
3	a_3	reply with letter	v_j replies to v_i's letter request, with the requested letter ℓ.
4	a_4	thinking	v_i is thinking.

Player use of alphabetic letters. Players use letters in different ways, depending on whether a letter is initially assigned, or has been received from a neighbor. A player has an infinite supply of their initially-assigned letters. This way, a player can share any of its n_ℓ letters with her distance-1 neighbors any number of times, to increase cooperation. At the same time, a player can use their own letters redundantly in any number of words. However, each letter that v_i receives from a distance-1 neighbor cannot be shared with other neighbors of v_i. Also, when a received letter is used to form a word, then this letter is consumed and hence can no longer be used. If v_i wishes to use that letter again, it must request and receive it again.

For example, if v_i has letters $L_i^{ih} = (e,s,t,m)$, and e and s are owned letters, then v_i can form the word *see* because e and s can be used any number of times in a word. After v_i submits *see*, it still has letters in-hand $L_i^{ih} = (e,s,t,m)$. However, if e is a received letter, then v_i cannot form *see*; v_i would need two e's from the same or different neighbors. Suppose v_i has $L_i^{ih} = (d,c,o,g,g,j)$ and that owned letters are (d,c,o). Then, when v_i forms *cog*, L_i^{ih} is updated to (d,c,o,g,j).

Forming words. A corpus of words C^W is provided for the game. A submitted word is considered valid if it is an element of the corpus. Otherwise it is invalid. A valid word may only be submitted one time by a player. The set of words formed up to time t by v_i is denoted W_i. W_i is the words formed over the entire game if no time is specified. However, multiple players can form the same word.

Illustrative cooperative actions in game play. A series of three time steps in a game is provided in Figure 1, from time t through time $(t+2)$. Time $(t+1)$ is separated into two visuals to make the dynamics clearer. There are four players, v_1 through v_4, arranged in a "circle" configuration where each node v_i has degree $d_i = 2$. Next to each player is a box containing letters. These represent the letters that a player has in-hand (i.e., in their possession) that can be used to form words. The letters that a player is assigned initially are the initial letter assignments, shown in the players' boxes of letters in black. For example, if we assume that $n_\ell = 3$ and that the letters in-hand at time t are the initial letters (also called owned letters), then v_1 has initial letters (u,r,k). Each player in the game knows her distance-1 neighboring players and her neighbors' letters so that they can be requested.

Figure 1 illustrates several actions in a game. At time t, v_2 thinks to form the word *bug* (magenta text) and therefore requests letter u from v_1. Player v_3 forms the word *cot* from its three owned letters, and hence maintains all three letters. At time $(t+1)$, v_1 responds to v_2 with letter u. Received letters are shown in brown to distinguish them from owned letters. Also at $(t+1)$, v_2 forms and submits word *bug* (that is why u was requested). (Blue text is an action.) v_2 loses u after forming *bug*. The game ends when $t = t_g$.

4 MODELS and ABM ALGORITHMS

4.1 Group Anagram Game

Figure 2 provides an abstract view, in the form of a graph $G(V,E)$, of the same game configuration as in Figure 1. Here, the four players $v_i \in V$, $i \in \{1,2,3,4\}$ are represented as graph vertices and the communication channels are represented by the edge set E, where $e = \{v_i, v_j\} \in E$ for $v_i, v_j \in V$. (Note that for models, we use the terms player, agent, node, and vertex interchangeably.)

Each agent $v_i \in V$ in the ABM is assigned a set of the data structures in Figure 2. See the caption for B_i^1 and B_i^2. Each entry in a buffer is a data structure itself. For example, B_i^2 contains letter requests, made by v_i. Fields in a request include: the letter ℓ requested, the particular word w that v_i seeks to form with the letters, the neighbor v_k to whom the request is sent, the requestor v_i, a unique universal identifier, and the time t of the request. Fields of letter requests in buffer B_i^1, to which v_i may reply (i.e., the reply buffer), include the requestor v_j, the letter requested ℓ, the universal identifier in the request that instigated this reply, the time t of the request, and the agent to which the request is made (which is v_i, used for verification). Another data structure (not shown) contains all letter requests made for a particular word w because a word may require multiple letters from neighbors. Let L_i^{ih} be the sequence of letters currently possessed by v_i in a GrAG. Let $L_i^{\prime}$ be the combined set of all letters initially assigned to all distance-1 (immediate) neighbors of v_i.

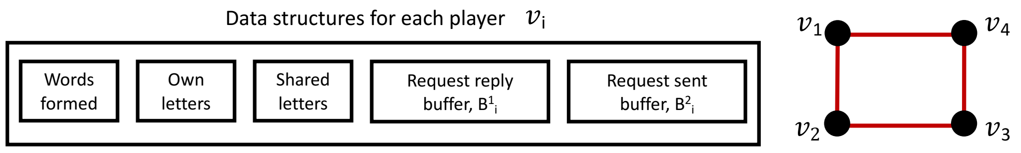

Data structures for each player v_i
Words formed
Own letters
Shared letters
Request reply buffer, B^1_i
Request sent buffer, B^2_i
v_1
v_4
v_2
v_3

Figure 2: Abstract view of the group anagram game (GrAG) (cf. Figure 1) to support modeling. Each node (player) $v_i \in V$ has the structures depicted to support the player actions in Table 1. Buffer B_i^1 contains letter requests from neighboring players to which v_i may respond. Buffer B_i^2 contains letter requests from v_i to its neighbors.

For each agent in each simulation, $p_{act} = (p_{fw}, p_{sreq}, p_{srep}, p_{think})$ is specified. This is a 4-tuple of fixed probability values, whose sum must be 1.0. These vector elements correspond, respectively, to the probability of an agent v_i taking actions a_1 through a_4 of Table 1 at each time step. At each time, each v_i uniformly samples a probability p^* from [0,1]. An agent deterministically chooses the action $a \in A$ based on p_{act} using p^*, and executes a.

Two algorithms are given in this section for modeling the GrAG. They are presented in a style for understanding both the agent model and the simulation process. Some details are omitted for clarity. For example, the software implementation uses a distributed HPC framework, but specifying aspects of the distributed computations provides no insights into the agent model nor the simulation process.

The controlling algorithm (not shown) loops over individual and independent executions of the game dynamics (each execution is called a *run*), such that a simulation is comprised of a user-specified number of runs. For example, multiple runs may be used to obtain variability for stochastic simulations where each run has the same initial conditions I. For each run, the simulator loops over discrete times, and for each time, the simulator iterates over all agents. For each agent, the VERTEX FUNCTION algorithm (see Algorithm 1) is invoked.

Algorithm 1 is the entrypoint into computations for individual agents $v_i \in V$. An agent first receives all messages from its neighbors that were sent in the previous time step, and buffers B_i^1 and B_i^2 are updated accordingly. Then, the action a that agent v_i takes at t is determined. The algorithm VERTEX ACTION (Algorithm 2) is invoked with v_i and action a. Algorithm VERTEX ACTION executes the action a. All actions are detailed in this algorithm, and may result in updates of buffers, forming a word, forming letter requests and replies, and sending a letter request or reply.

Recall that letters that players have available to form words are used differently, depending on whether the letter is initially assigned to a player or is a letter of a distance-1 (immediate) neighbor. A letter that is initially assigned to v_i can be used any number of times, including multiple times within one word. So a player can form *tot* with initial letters $L_i^{init} = (b,t,o)$, but if v_i's initial letters are $L_i^{init} = (b,g,o)$, then v_i must request and receive two t's from neighbors.

Algorithm 1: Steps of the Algorithm VERTEX FUNCTION.

```
1 Input: Time $t$. Agent $v_i$. Neighbors $n[i]$ of $v_i$ in $G(V,E)$. Letters $L_i^l$ of neighbors of $v_i$. Letters that $v_i$ has in-hand $L_i^{ih}$. Probabilities of
    actions in $A$, $p_{act} = (p_{fw}, p_{sreq}, p_{srep}, p_{think})$. Word corpus $C^W$. Buffer of $v_i$'s outstanding letter requests made $B_i^2$. Buffer of letter
    requests to $v_i$, $B_i^1$.
2 Output: The next action $a \in A$ and the updated values (state) of all inputs.
3 Steps:
    A.  Receive all letter requests from neighbors $n[i]$ of $v_i$, sent to $v_i$ at the previous time $(t-1)$, and put in buffer $B_i^1$. These are requests that $v_i$
        may reply to.
    B.  Receive all letter replies from $v_i$'s neighbors that are in response to $v_i$'s letter requests, sent to $v_i$ at the previous time $(t-1)$, and put in
        $L_i^{ih}$; mark this letter request in $B_i^2$ as fulfilled. If received letters enable a visited word to now be formed, then form the word and submit.
    C.  Sample from a uniform distribution [0,1], value $p^*$.
    D.  From the vector $p_{act} = (p_{fw}, p_{sreq}, p_{srep}, p_{think})$ of probabilities for actions $a_1$ through $a_4$, respectively, of Table 1, and $p^*$, determine the
        action $a \in A$ that $v_i$ will take at this time $t$.
    E.  Call the action routine for vertex $v_i$ (Algorithm 2, VERTEX ACTION).
    F.  Write updated state (variable values) to file for $v_i$ at time $t$.
```

```
Algorithm 2: Steps of the Algorithm VERTEX ACTION for vertex $v_i$.
1 Input: Time $t$. Agent $v_i$. Action $a \in A$. Neighbors $n[i]$ of $v_i$ in $G(V,E)$. Letters $L_i'$ of neighbors of $v_i$. Letters that $v_i$ has in-hand $L_i^{ih}$.
    Probabilities of actions in $A$, $p_{act} = (p_{fw}, p_{sreq}, p_{srep}, p_{think})$. Word corpus $C^W$. Buffer of $v_i$'s outstanding letter requests made $B_i^2$.
    Buffer of letter requests to $v_i$, $B_i^1$.
2 Output: The updated values (state) of all inputs.
3 Steps:
    A.  if $a$ equals $a_1$ do ## Action $a$ is for $v_i$ to form word.
            i. Select randomly a word $w \in C^W$, where $w \notin W_i$ (i.e., $v_i$ cannot repeat words), and can be formed with letters in $L_i^{ih} \cup L_i'$ (i.e., the
               union of letters in-hand and the letters of neighbors). If all letters, including multiplicities, of $w$ are in $L_i^{ih}$ (i.e., are in-hand), then
               form and submit word. Otherwise, there are letters $\ell$ that need to be requested. Put these letters $\ell$ of $w$ (each needed letter instance
               is an individual request) in $B_i^2$ (to be requested). If there is no such $w \in C^W$, do nothing.
    B.  if $a$ equals $a_2$ do ## Action $a$ is for $v_i$ to send letter request.
            i. If there is a letter request in $B_i^2$ that has not been sent, send one request using first-in first-out (FIFO) order. Mark request as sent.
               Otherwise, do nothing.
    C.  if $a$ equals $a_3$ do ## Action $a$ is reply (with letter) to a letter requested by a neighbor.
            i. If there is a letter request from a neighbor of $v_i$ in $B_i^1$, that is waiting to be fulfilled, then send a letter reply using FIFO ordering,
               and mark the request in $B_i^1$ as fulfilled. Otherwise, do nothing.
    D.  if $a$ equals $a_4$ do ## Action $a$ is think.
            i. Do nothing. The process of $v_i$ thinking just consumes time.
    E.  Return updated state (variable values) for input variables (above) of vertex $v_i$.
```

4.2 Performance Parameters For Individual Players

Let $L_i = L_i^{ih} \cup L_i'$ be the sequence of all letters available to v_i (its own assigned letters and those of its neighbors). The set $W_i^{tot} \subseteq C^W$ of words that v_i can form are the words $w \in C^W$ such that every letter ℓ of w is in L_i.

The performance of v_i in a GrAG is given by three parameters: (i) $\alpha_{w,i}$ in forming words, (ii) $\alpha_{req,i}$ in requesting letters, and (iii) $\alpha_{rpl,i}$ in replying to letter requests of its neighbors, given by

$$\overbrace{\alpha_{w,i} = \frac{n_{words,i}}{n_{words,i}^{max}}}^{\text{words formed}}, \qquad \overbrace{\alpha_{req,i} = \frac{n_{reqSent,i}}{n_{reqSent,i}^{max}}}^{\text{letter requests sent}}, \qquad \overbrace{\alpha_{rpl,i} = \frac{n_{replSent,i}}{n_{reqRec,i}}}^{\text{letter replies sent}} \tag{1}$$

where $n_{words,i}$ is the number of words that v_i forms in the GrAG; $n_{words,i}^{max}$ is the maximum number of words that v_i can form, i.e., $n_{words,i}^{max} = |W_i^{tot}|$; $n_{reqSent,i}$ is the number of letter requests that v_i sends to its neighbors in $G(V,E)$ in the GrAG; $n_{reqSent,i}^{max}$ is the maximum number of requests that v_i can send to all of its immediate neighbors in $G(V,E)$ in order to form all words in W_i^{tot}; $n_{replSent,i}$ is the number of letter replies sent by v_i to its neighbors; and $n_{reqRec,i}$ is the number of letter requests received from v_i's neighbors.

5 MODELING AND SIMULATION RESULTS

5.1 Considerations for Simulation Parameters

Table 2 provides the simulation variables studied. Here, variables are discussed in relation to practical (realistic) considerations for implementing the game we are modeling. The number n of players on the lesser end ($n = 11$) represents smaller teams, and larger n ($= 1000$) simulates a game that is possible to play, for example, by employing students from a large undergraduate course that can be on the order of many hundreds of students. Networks are used to control player degree (number of neighbors) in order to scale up n. For example, it is not practical for $n = 100$ students to all interact with each other.

The use of circle, clique, and random regular graphs means that, for a specific graph, all nodes (agents, players) have the same number of neighbors. Hence, each node is, in a sense, a different replicate instance, because letter assignments vary among the nodes but the numbers of letters assigned to each node in these simulations is the same.

For the probability vector p_{act} of action probabilities for a player, we use $\hat{p}$ values in Table 2; $\hat{p}$ corresponds to probabilities for actions form word, request letter, and reply to requests. Values range from

0.05 to 0.25, to represent a range from inactive (or more deliberate) agents to relatively active (fast thinking) agents. This is, when $\hat{p} = 0.05$, $p_{think} = 0.85$, so that in expectation, a player is thinking—not acting—for 85% of the game time. The mid-range value of $\hat{p} = 0.15$ means that a player is thinking more than one-half of the time ($p_{think} = 0.55$), and the greatest value for $\hat{p}$ means that an agent is only thinking 25% of the time and acting 75% of the time. We study this range to evaluate different levels of mental activity in lieu of a cognitive model of player behavior.

The numbers n_ℓ of own letters assigned to players is designed to control interactions (along with node degree d). The cases $n_\ell \leq 2$ force players to interact with their neighbors in order to form words when the minimum permissible word length is three letters. A player has access to as many as $(d+1)n_\ell$ unique letters (if there are no duplicate letters), if all players are given n_ℓ letters. Owned letter assignments are done such that each player has n_ℓ unique letters; there are no duplicates.

The word corpus of 1015 words considers only 3-letter words. Admitting larger words is a subject for further work, but is unlikely to effect our results, at least qualitatively, because of the way agents step through C^W to select words to try to form.

Table 2: Summary of parameters and their values used in simulations of GrAGs.

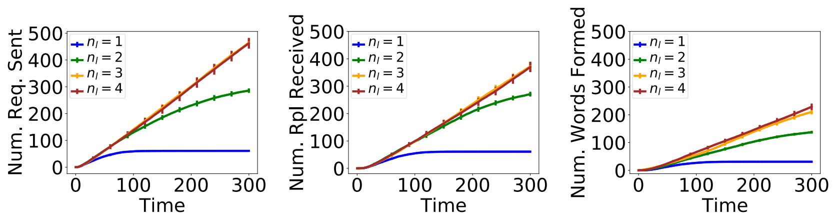
Parameter	**Description**
Networks $G(V,E)$.	Circle graphs, cliques, and random regular graphs. Numbers of nodes are $n = 11$, 100, and 1000. Uniform degrees d of players are two and ten.
Probability vectors for actions p_{act}.	$p_{act} = (p_{fw}, p_{sreq}, p_{srep}, p_{think})$ is the vector of probabilities corresponding to the actions in Table 1 where the probabilities associated with a_1, a_2, a_3, and a_4 are, respectively p_{fw}, p_{sreq}, p_{srep}, and p_{think}. These probabilities are used at each time t to select the player action at that t. We use $\hat{p} = p_{fw} = p_{sreq} = p_{srep}$, where $\hat{p}$ is in the set $\{0.05, 0.10, 0.15, 0.20, 0.25\}$. Consequently, $p_{think} = 1 - 3\hat{p}$. A vector p_{act} of values is assigned to each agent.
Number n_ℓ of owned letters.	This is the number of owned letters assigned to a player as part of initial conditions. Values used are $n_\ell = 1$, 2, 3, and 4. Owned letters for a player purposely contain no duplicates. A value of n_ℓ is assigned to each player.
Letter assignment process.	The initial letter assignments to players are done uniformly at random.
Shared letters.	The letters that a player can share with her neighbors are the same as the owned letters. There may be duplicate letters between pairs of players, including neighbors of an agent v_i.
Duration of GrAG t_g.	The duration of the group anagram game is fixed at $t_g = 300$ seconds.
Word corpus C^W.	The corpus of 1015 3-letter words is taken from *http://www.wordfind.com/3-letter-words/*. That is, only 3-letter words are considered in simulations.
Number of runs n_{runs}.	Each simulation is composed of n_{runs} individual dynamics instances, where each instance starts from time $t = 0$, with initial conditions reset, and then the dynamics of the system are executed for t_g discrete time steps. Here, $n_{runs} = 50$.

5.2 Simulation Results for GrAG

Subsections below provide insights into the effects of input variables (see Table 2) on simulation results. Results include all 50 runs (instances) per simulation, either in the form of averages (i.e., the mean values over all runs at each discrete time) with $\pm$ one standard deviation error bars, or boxplots of all data. Observations of behaviors hold only for the conditions of the computational experiments (but are suggestive of more general trends).

Aggregate effects of number of owned letters. Figure 3 shows aggregate time histories for all agents in an 11-agent game where each agent has two neighbors, i.e., $(n,d) = (11,2)$. Other simulation parameter values are given in the caption. The numbers of letter requests sent by players are in Figure 3a. The numbers of replies (with the letters) to those requests (Figure 3b) are a little less than the numbers of requests. The numbers of words formed (Figure 3c) are still less in number than requests and replies.

The results indicate that there is not much difference between giving players three or four initial letters. However, the differences for $1 \leq n_\ell \leq 3$ are large. Also, $t_g = 300$ seconds is more than adequate for $n_\ell = 1$ because performance saturates, but for $n_\ell \geq 2$, performance is still changing, although it is approaching saturation for $n_\ell = 2$.



(a) (b) (c)

Figure 3: Simulation results for all agents from an $(n,d) = (11,2)$ graph, for action probabilities $p_{act} = (p_{fw}, p_{sreq}, p_{srep}, p_{think}) = (0.15, 0.15, 0.15, 0.55)$ and number of letters n_ℓ taking values $\{1,2,3,4\}$ in turn (see legend). Standard deviation error bars are shown in each curve, every 30 time units, so as to not overwhelm the curves. Plots (left to right) correspond, respectively, to actions: (a) send (letter) requests, (b) receive (letter) replies, and (c) form words. The ordinate values are the *sum* of all of the respective actions across all 11 agents, so the average number of agent actions is obtained by dividing each ordinate value by 11. Among the results shown are: when a player has only one letter ($n_\ell = 1$), the actions saturate in less than 150 seconds, but for greater numbers of owned letters, actions are taking place at the 5-minute mark; and the differences between results for $n_\ell = 3$ and 4 are small.

Effect of number of neighbors. Figure 4 shows aggregate results for two different values of numbers of neighbors, for experiments with $n = 11$. For all graphs, curves for $d = 2$ are in blue and those for $d = 10$ are in green. For each degree, the solid curves correspond to $n_\ell = 2$, and the dashed curves correspond to $n_\ell = 4$. Where no dashed curve is visible, it is "under" (coincident) with a solid curve.

These results are particularly interesting from the perspective of agents' "congestion" of communications and performance. For agent v_k that is replying to requests with the desired letters, if v_k has degree $d = 2$, then all of her replies are going to two agents v_1 and v_2. If v_k has $d = 10$ neighbors, then the replies are being distributed across more neighbors. The main point that the plots convey is that players with large numbers of neighbors can get bombarded with letter requests. That is, when d of a player v_k is sufficiently large, p_{srep} for v_k is sufficiently small, and p_{sreq} of v_k's neighbors are sufficiently large, then the letter requests made of v_k "pile up" because v_k cannot reply sufficiently quickly to these requests. The result is that performance, in terms of words formed, can *decrease* as d increases. This is why, in Figure 4, the green curves (for $d = 10$) in Figure 4a for letter requests sent are at or above the blue curves (for $d = 2$), but the green curves fall below the blue curves in Figures 4b and 4c for replies received and words formed, respectively.

Performance of individual (disaggregated) players. Figure 5 disaggregates the end-of-game ($t_g = 300$ seconds) results for $n_\ell = 3$ from Figure 3, and compares player actions with optimal behavior—in terms of numbers of actions. *Optimal behavior* means the best possible performance of players. See Section 4.2 for definitions of performance ratios. Performance in terms of formed word $\alpha_{w,i}$, letter requests *sent* $\alpha_{req,i}$, and letter replies *sent* $\alpha_{rpl,i}$ are in the range $[0,1]$ in Figures 5a, 5b, and 5c, respectively, per player. There are clear similarities between $\alpha_{w,i}$ and $\alpha_{req,i}$, since letter requests (and replies) will make it possible to form more words. Figure 5d makes it clear that the relatively good performances in words formed and letter requests for Players 0 and 4 are due to the relatively lesser optimal behaviors possible (red and black curves for these players [for requests sent and words formed, respectively] have lesser values). The blue curve, which serves as the denominator for the data in Figure 5c, also explains these data. These data,

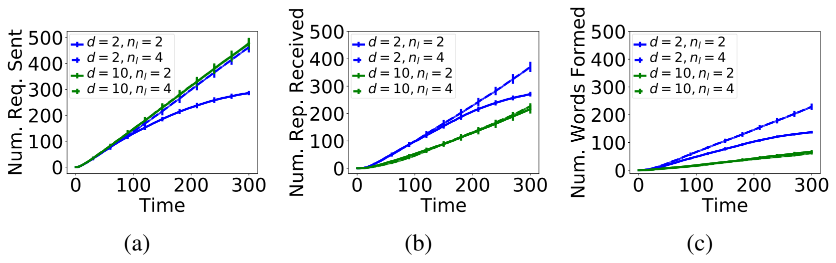


Figure 4: Simulation results aggregated for all agents for graphs with $(n,d) = (11,2)$ (blue curves) and (11, 10) (green curves), for action probabilities $p_{act} = (p_{fw}, p_{sreq}, p_{srep}, p_{think}) = (0.15, 0.15, 0.15, 0.55)$, and number of letters $n_\ell = 2$ (solid curves) and 4 (dashed curves). Standard deviation error bars are shown in each curve, every 30 time units, so as to not overwhelm the curves. Plots (left to right) correspond, respectively, to actions: (a) send (letter) requests, (b) receive (letter) replies, and (c) form words. The ordinate values are the *sum* of all of the respective actions across all 11 agents, so the average number of agent actions is obtained by dividing each ordinate value by 11. Taken together, the plots show the interesting result that players that have too many neighbors (so that they have more letters to select from to form words) receive fewer letter replies and form fewer words. See the text for details.

collectively, indicate that initial letter assignments can make a big difference in performance parameters $\alpha_{w,i}$, $\alpha_{req,i}$, and $\alpha_{rpl,i}$.

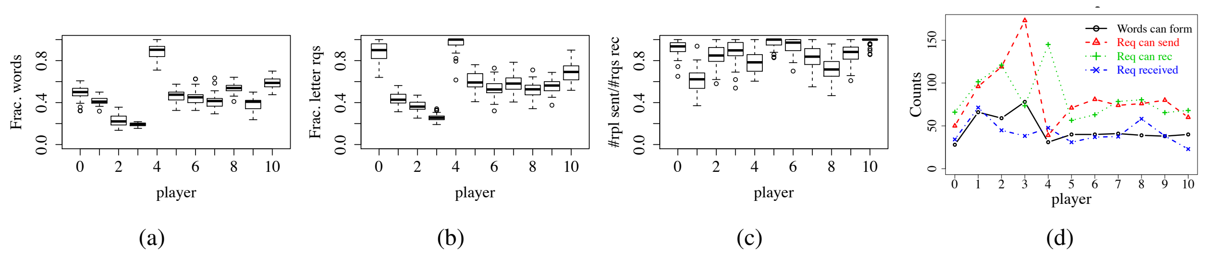


Figure 5: Performance results of individual agents relative to best possible performance for an $(n,d) = (11,2)$ graph, action probabilities $p_{act} = (p_{fw}, p_{sreq}, p_{srep}, p_{think}) = (0.15, 0.15, 0.15, 0.55)$, and number of letters $n_\ell = 3$ assigned to each player. Plots (left to right) correspond, respectively, to actions: (a) $\alpha_{w,i}$ for words formed, (b) $\alpha_{req,i}$ for letter requests sent, and (c) $\alpha_{rpl,i}$ for replies sent for letter requests. In all cases, boxplots show results for all 50 run instances. (d) Best possible performance in terms of numbers of actions for forming words, requests sent, replies sent ($\equiv$ requests received), and actual requests received. The black and red curves show, for example, that player 4's strong performance is due to the fact that it can form relatively fewer words and request relatively fewer letters than most other players.

Average performance across agents for changes in action probabilities. Figure 6 shows the effects of different player activity levels, implemented by altering $\hat{p}$ in p_{act}; see Table 2. As $\hat{p}$ increases, the probabilities of each of the actions form word, request letter, and reply to request increase. The figure shows that the probability (i.e., activity) of players has a significant effect on performance, as does the number n_ℓ of letters assigned to players.

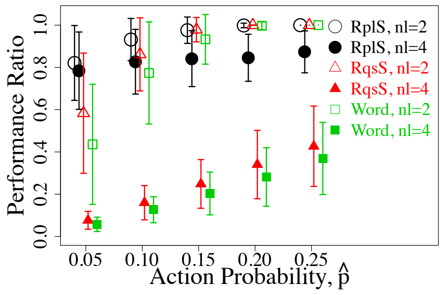


Figure 6: Average performance across all players in $(n,d) = (11,2)$ experiments for $n_\ell = 2$ and 4, and for all p_{act} vectors with $\hat{p} = 0.05$, 0.10, 0.15, 0.20, and 0.25 (see Table 2). Y-axis values are average values of $\alpha_{w,i}$ for form words ("Word" in green), $\alpha_{req,i}$ for letter requests sent ("RqsS" in red), and $\alpha_{rpl,i}$ for replies sent for letter requests ("RplS" in black). Player performance increases as $\hat{p}$ increases and n_ℓ decreases.

6 USES OF MODELING RESULTS

Simulations results of the previous section are used to discuss refinements to game parameters of the envisioned experimental platform and to construct hypotheses to test.

6.1 Game Specifications

Several insights from the simulation results above can be used to specify/refine game conditions. We assume that the probabilities of player actions form word, request letter, and reply to request are in the middle of our investigated range, i.e., $p_{act} = (p_{fw}, p_{sreq}, p_{srep}, p_{think}) = (0.15, 0.15, 0.15, 0.55)$. Figure 3 suggests that a 5-minute game may be adequate for players to form words when $n_\ell \geq 3$ (that is, individual players may form up to 20 or more words). This is not an obvious result. Individual anagram games of a different sort, where a player unscrambles letters to form a *single* unique word, use a duration of four minutes per game (Mayzner and Tresselt 1958). Figure 3, where $d = 2$ and $n_\ell = 3$, suggests that a criterion on d and n_ℓ to produce significant interactions among players is: $(1+d)n_\ell \geq 9$. Results in Figure 4 provide strong evidence for the need to test different network structures. While some may contend that this is an obvious statement, this figure suggests *why* testing different connectivities is important and interesting. Figure 5 indicates that (initial) letter assignments can produce different performance among players. We saw that for Players 0 and 4, performance was aided by the fact that they could form fewer words and could profitably request fewer letters than other players. This is dictated by assignments of a player's and its neighbors' letters. (Our ABMS can control the assignment of particular letters to particular players, but this is not reported on here owing to lack of space.) Although not shown, results for different n ($= 11$, 100, and 1000) players show that numbers of actions scale linearly with n, as expected.

6.2 Illustrative Hypotheses

From the computational results of Section 5, the hypotheses in Table 3 were formulated. These could be tested in a game platform that runs experiments according to the description in Section 3. Conditions not stated in the hypotheses are those represented in the figures of Section 5. For the last one, we did not show results for games with $n = 100$ and 1000 players owing to space limitations. Finally, the hypotheses are quantitative rather than qualitative to provide more stringent tests.

7 CONCLUSION AND LIMITATIONS

We have described a novel GrAG. Novelty and contributions are in Sections 1.3 and 1.4. Future work includes addressing limitations by extending the model to account for how players decide on which actions to take. We need to report on simulations with heterogeneous conditions. Space limitations prevent presentation of these results. Source code is available upon request from Chris Kuhlman at cjk8gx@virginia.edu.

Table 3: Illustrative hypotheses, formulated from the modeling results, to test in an experimental setting.

Topic	**Hypothesis**
Game duration.	A five-minute GrAG is sufficiently long to produce over 1000 total actions among players in $(n,d) = (11,2)$ experiments when players have $n_\ell = 3$ letters.
Numbers n_ℓ of assigned letters.	Increasing n_ℓ from 3 to 4 results in no more than a 10% increase in numbers of actions.
Player degree d.	As number of neighbors of a player increases from $d = 2$ to 10, numbers of letter requests increases by 100% but the number of letter replies decreases by 100%.
Numbers of players.	As the number n of players in the game increases from 11 to 100, the numbers of actions will increase disproportionately by 20%.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their helpful feedback. We thank our colleagues at NSSAC and computer system administrators: Dominik Borkowski, Jason Decker, Miles Gentry, Jeremy Johnson, William Marmagas, Douglas McMaster, and Kevin Shinpaugh. This work has been partially supported by NSF CRISP 2.0 (Grant 1832587), DARPA Cooperative Agreement D17AC00003 (NGS2), DTRA CNIMS (Contract HDTRA1-11-D-0016-0001), DTRA Comprehensive National Incident Management System Contract HDTRA1-17-D-0023. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

REFERENCES

Cadsby, C. B., F. Song, and F. Tapon. 2007. “Sorting and Incentive Effects of Pay for Performance: An Experimental Investigation”. *Academy of Management Journal* 50:387–405.

Charness, G., R. Cobo-Reyes, and N. Jimenez. 2014. “Identities, Selection, and Contributions in a Public-Goods Game”. *Games and Economic Behavior* 87:322–338.

Feather, N. T., and J. G. Simon. 1971. “Causal Attributions for Success and Failure in Relation to Expectation of Success Based Upon Selective or Manipulative Control”. *Journal of Personality and Social Psychology* 39:527–541.

Mayzner, M. S., and M. E. Tresselt. 1958. “Anagram Solution Times: A Function of Letter Order and Word Frequency”. *Journal of Experimental Psychology* 56(4):376.

Ren, Y., V. Cedeno-Mieles et al. 2018. “Generative Modeling of Human Behavior and Social Interactions Using Abductive Analysis”. In *ASONAM*, 413–420.

Russell, D. G., and I. G. Sarason. 1965. “Test Anxiety, Sex, and Experimental Conditions in Relation to Anagram Solution”. *Journal of Personality and Social Psychology* 1:493–496.

Yang, Y., L. Mao, and S. S. Metcalf. 2019. “Diffusion of Hurricane Evacuation Behavior Through a Home-Workplace Social Network: A Spatially Explicit Agent-Based Simulation Model”. *Computers, Environment, and Urban Systems* 74:13–22.

AUTHOR BIOGRAPHIES

ZHIHAO HU, **VANESSA CEDENO-MIELES**, **PARANG SARAF**, are graduate students at Virginia Tech. Their email addresses are huzhihao,vcedeno,parang@vt.edu.

XINWEI DENG is a professor in the Department of Statistics at Virginia Tech. His email address is xdeng@vt.edu.

ABHIJIN ADIGA, **GIZEM KORKMAZ**, **CHRIS J. KUHLMAN**, **DUSTIN MACHI**, **MADHAV V. MARATHE**, **S. S. RAVI** are faculty in the Biocomplexity Institute & Initiative at University of Virginia. Their email addresses are abhijin,gkorkmaz,cjk8gx, dm8qs,marathe,ssr6nh@virginia.edu.

YIHUI REN is a Research Associate at Brookhaven National Laboratory. His email address is yren@bnl.com.

SALIYA EKANAYAKE is a Postdoctoral Fellow at Lawrence Berkeley National Laboratory. His email address is esaliya@gmail.com.

BRIAN J. GOODE, **NAREN RAMAKRISHNAN**, **NATHAN SELF** are, respectively, faculty in the Biocomplexity Institute of Virginia Tech, director of the Virginia Tech’s Discovery Analytics Center (DAC), and faculty in DAC at Virginia Tech. Their email addresses are bjgoode,naren,nwself@vt.edu.