Computational Social Networks

RESEARCH Open Access

Robust communication network formation: a decentralized approach

Christopher Diaz[1], Alexander Nikolaev[1], Abhinav Perla[1*], Alexander Veremyev[2] and Eduardo Pasiliao[3]

*Correspondence: aperla@buffalo.edu
[1] Department of Industrial and Systems Engineering, University at Buffalo, Buffalo, NY, USA
Full list of author information is available at the end of the article

Abstract
The formation of robust communication networks between independently acting agents is of practical interest in multiple domains, for example, in sensor placement and Unmanned Aerial Vehicle communication. These are the cases where it is only feasible to have the communicating actors modify the network locally, i.e., without relying on the knowledge of the entire network structure and the other actors' decisions. This calls for approaches to optimizing network structure in a decentralized way. We present an actor-oriented modeling approach to design and parameterize models that enable the creation of networks that exhibit the properties desirable for efficient information sharing. Computational experiments show that the achieved network formation rules, specified in a calculated way, allow agents to maintain robust network structure by activating only a limited number of direct communication channels. The obtained results are promising, as evidenced by the reported comparisons to optimal network configuration solutions obtained in a centralized way.

Keywords: Network optimization, Decentralized optimization, Social networks, Actor-oriented modeling, Mixed-integer programming, Nelder–Mead algorithm

Introduction

The formation of a communication network is an optimization problem of considerable practical interest. It is the problem of organizing a group decision-making process with a desirable final outcome—a formed network—under a set of constraints that typically include a (hard or soft) limit on communication capacity, or the number of connections that each agent could maintain. A group decision-making process describes (explains) how each agent must behave, i.e., act to build and/or sever communication channels in its own locality. Centralized group decision-making processes, however, are not always adequate for the task at hand, e.g., when dealing with applications related to sensor networks [1], or when Unmanned Aerial Vehicles need to establish and maintain communication, while either not having access to a central hub or preferring not to use this direct transmission channel [2].

Social network formation principles can play a useful role in the modeling of opinion/belief spread en route to achieving the network outcomes. These principles can define how the agents observe their neighbors' actions and local connection structure over time and, based on this information, make better judgments towards what actions could potentially lead to the best global network outcome [3, 4]. Some research in the domains of network analysis and communication exchange addresses the need to efficiently spread information throughout a network [5, 6]. However, while multiple author groups examine the topic of modifying structural network properties in a centralized manner [7–9], only one general social network analysis approach exists to model decentralized decision making; this calls for extending the utility of this approach, called actor-oriented modeling [10], beyond studying friendship network formation as it was originally done, towards building networks with desirable properties for efficient information exchange, in a decentralized manner. Specifically, in the language of social network analysis, the actors modeled in this paper work to optimally position themselves to hear and forward rumors. Each actor is assumed to have limited information about its own embeddedness into a communication network. In the network formation stage, this information is supplied to each actor periodically via the messages that reach the actor from and through their peers. Each actor then makes its own local decisions, i.e., decides to maintain only certain ties in each formation period, eventually stopping at a desirable network outcome.

This paper makes several contributions *en route* to formulating and testing a framework for organized communication network formation. First, this paper describes a decentralized deterministic information propagation (DDIP) network formation model, where actors link to each other (using or not using available communication channels) in a decentralized manner and, in doing so, build a network that provides each actor with an ability for robust reception and propagation of information across the network. Second, this paper makes advances to enable calculated parameterization of DDIP models for sustained and reliable exchange of information. Here, in addition to studying DDIP model outputs—the formed networks—using discrete-event simulation, we employ Markov Chain theory to replace the *message-passing* component in these simulation runs by more accurate and much faster analytical analyses. This modeling advance allows us to run an iterative model parameterization algorithm that executes in reasonable time. Third, the results of this venture are assessed in the testing and evaluation stage, where the networks produced by the proposed optimized DDIP model are systematically compared against the centralized network formation solutions. The centralized solutions are obtained using Mixed Integer Programs that find optimal communication network structures given the original layout, i.e., feasible communication channels, of any network.

The remainder of this paper is organized as follows. "Practical problem setting and relevant literature" provides additional practical motivation for our problem formulation, followed by a literature review which describes the prior research done in the area of decentralized communication network analysis. "The decentralized deterministic information propagation model for network formation" details the DDIP model. "Parameter estimation" describes an approach to evaluating the DDIP model performance and formulates a parameter search optimization problem to tune up the model to be useful in specific communication settings that can be realistically useful. "Computational experiments" describes the generation of synthetic datasets and the design of experiments to run the parameter search optimization on, and then, reports the computational results obtained, comparing them to the results obtained from executing the specially formulated mixed integer programs that achieve centrally optimal solutions for the considered problem instances. "Conclusion" summarizes the findings, offering conclusions and discussion about the potential future directions of this research.

Practical problem setting and relevant literature

Consider a sensor network (or, more generally, a network of actors) comprised of a group of sensors (actors) that are linked by a wireless medium and tasked to perform distributed sensing, decentralized decision making and/or inference tasks. In such networks, sensors (actors) gather information about the physical world, conduct analyses based on local information, and then perform the ensuing actions upon the environment, thereby enabling automated and remote interaction with the environment.

In the above, the meaning of term "actor" differs from the conventional notion where an object is restricted to "act" in a particular way. Instead, the said actor, besides being able to act continuously on the environment, is also a network entity that performs networking-related functions, i.e., those including receiving, transmitting, processing and relaying information. For example, a robot may interact with the physical environment; however, from a networking perspective, the robot constitutes a single entity that is referred to as actor. Hence, the term actor can embrace heterogeneous devices including robots, unmanned aerial vehicles and networked actuators.

Applications of wireless sensor and actor networks may include teams of mobile robots that perceive the environment from multiple different viewpoints based on the data gathered by sensor networks. However, due to the presence of actors (that are usually resource-rich devices equipped with better processing capabilities, stronger transmission powers and longer battery life), the wireless sensor and actor networks differ from wireless sensor networks [1]. Moreover, in the wireless sensor and actor networks, depending on the application, there may be a need to rapidly respond to external sensor data input/flow. To successfully inform actions, sensor data must still be valid at the time of acting. Therefore, the issue of real-time communication is very important in the wireless sensor and actor networks, since actions are often required to be performed on the environment immediately after the sensing takes place [11].

The number of sensor nodes deployed in typical conventional applications may be in the order of hundreds or thousands. However, such a dense deployment is not necessary for "actor" actors due to the different coverage requirements and physical interaction methods supporting the acting tasks. Hence, in the wireless actor networks, the number of actors is much lower [1]. Therefore, in order to provide effective sensing, coupled with acting or inference tasks, a distributed local coordination mechanism is necessary among sensors/actors. Building such a distributed coordination mechanism in uncertain environments requires broader studies, which examine actor-centered and actor-based network models. This calls for reviewing the existing works on information dispersion and decentralized communication networks and also the works from the social network analysis domain that can be useful in the context of decentralized communication.

There have been a number of works published recently that focus on how information flows through networks. Currently, much of the research in this field focuses on how to maximize the spread of influence across networks [12] and also on how to form teams [13]. Of greater relevance to this paper, however, are the works within the realm of information propagation and diffusion that do not focus on maximizing or describing the effects of influence across networks. In [6], several models are proposed to only explain how information spreads. In [14], a partially absorbing random walk is employed to model how information passing can be used to learn the structure of a network. In [15], a transduction process, also employing absorbing random walks, is developed to traverse graphs. In [7], an evolutionary game theoretic framework is offered to predict information diffusion across networks.

The literature on decentralized methods to improve communication efficiency is also relevant to this paper, particularly because it emphasizes network robustness. Most of the existing papers on decentralized methods for network modeling focus on achieving consensus in a network in an organized manner [16]. The authors of [5] create sets of decentralized conventions that focus on forming consensus throughout a network. In [17], a more general framework is presented for analyzing multi-agent systems that enables the actors to reach consensus. In [18], the use of decentralized communication networks is analyzed with respect to controlling autonomous actors.

There is a gap in the body of literature that pertains to decentralized communication networks, as most papers are focused on how to create consensus in processing information, as opposed to looking at information passing and building robust communication networks in the first place. This paper aims at filling this gap by relying on the recent advances in social network analysis.

The use of models to analyze and predict the formation of social networks is a large topic of research. Descriptive social network techniques were used in [19] to create new ways to manage crisis de-escalation approaches. The use of social network analysis with respect to engineering and construction project management is evaluated in [20]. In [21], a model is developed to predict how groups of unconnected agents form social networks, focusing on how the pattern of interaction between agents causes networks to form. In [22], the evolution of social and economic networks is studied, where the dynamics of the individuals that make up the network are of particular importance. The actor-oriented modeling ideas for social network formation are described, e.g., in [23] and [10]. In the latter, Snijders et al. introduced a model, in which individuals periodically make changes to their local networks, with each maximizing their own weighted "happiness function" value. There has been some research in the use of decentralized methods to ensure network connectivity, as seen in [24], which focuses on the use of a decentralized control algorithm to balance the edges of a network. The latter work, however, focused mainly on making sure that connectivity was preserved and did not place emphasis on the information passing properties of the resulting graphs; the model presented in our paper fills this gap.

The decentralized deterministic information propagation model for network formation

The decentralized deterministic information propagation model (DDIP model) for network formation, introduced in [25] and much more rigorously studied in this paper, prescribes a particular order of communication network evolution, i.e., change in its structure, over multiple time periods. In each period, all the actors participate in several network building-related activities, performed sequentially in stages.

The actor-oriented modeling approach, conceived by Snijders [10] for social network analysis, provides a basis for many of the concepts used in the DDIP network formation model. In the DDIP model, each actor possesses their own personal objective function, which is a concept that was introduced in actor-oriented modeling. However, in the latter, an actor evaluates their personal "happiness" based purely on local graph structures. This differs in the DDIP model, as the actor's "happiness" must inform the agent of their location with respect to far-away peers as well and, hence, is designed to be a function of the fraction of messages it can receive from all the other actors in the network.

Another element of the DDIP model that is new compared to original actor-oriented models is the form of the weights defining the objective function for each actor. In the actor-oriented model [10], each network structure property (*aka.* "network effect" or "structural signature") used in each actor's objective function is weighted by the given parameters, and these parameters do not change dynamically as the network is formed. The DDIP model also uses weights in the objective equation, but these weights are used to place preference on the reception of messages from each other actor in the network. This preference is updated in every period per our model and is dependent on the proportion of messages each actor receives in each period.

Having established the degree of its reference to prior work, let us now review the DDIP model stages. The sequentially executed network formation activities start with a *message forwarding* stage—each actor forwards out messages to their connected network neighbors following an absorbing random walk. The message forwarding stage is crucial as it enables each actor to update its belief about its own position with respect to the rest of the network, in a decentralized way, using the information derived from the fraction of the total messages received from all other actors in the network. Next is the *weight and objective calculation* stage—each actor passes to its neighbors the information regarding the fraction of all messages it received, so that all actors update their beliefs about how well connected they are: specifically, these updates lead to the revisions of the actors' own weight functions and to the recalculation of their own objectives. Final is the *network estimation and modification* stage—actors estimate how the changes to their local network structures would affect their unique objective functions. The changes for each actor could potentially lie in either add or drop one incoming tie, or keep the present status-quo. Actors look to make such changes in the local connection structure that lead to the largest increase in the reception of messages from all other actors in the network, with a special emphasis placed on the messages from their estimated far-away peers, similar to the way actors make changes in [10]. After all the actors have acted in a single time period in the network estimation and modification stage, the next period begins, and the message forwarding and update procedures are repeated, until a preset stopping criterion is met. Such a criterion may be based on the amount of time, or time periods, allocated for the network formation as a global process.

The following subsections provide more rigor in the DDIP model definition, stating specifically how the employed variables and parameters affect the network building process, and how each variable is updated stage by stage.

DDIP model basics

Consider a set of actors $V = \{1, 2, \ldots, n\}$, where each actor corresponds to a node of a directed communication network, deployed over a region that needs to be monitored. The key motivation here is that the information gained through sensing the region needs to be broadcast to rest of the actors, using a directed communication network explicitly set up for this purpose, to take mission-related decisions.

Let P_i denote the set of actors which are in the communication transmission range of actor i and can potentially exchange information with actor i: this set refers to what peers are in the immediate *proximity* of actor i. To exchange sensed information, the network actors form directed ties with other actors in their proximity, where a directed tie is only possible between actors in proximity. Some actors are thus assumed to not be able to connect with certain peers, consistent with proximity restrictions and reflecting the fact that some actors may be too far away from each other to communicate directly. A directed edge between two nodes indicates that information can be transmitted from one actor to another: specifically, an *incoming* directed tie $i \leftarrow j$ signals that actor i has actor j as in-directed neighbor and is in a position to receive information from j.

It is worthwhile to discuss the justification for the assumption of the directed nature of the communication network being formed. While an undirected network is perfect under a centralized setting, it is impractical under a decentralized setting, as, when an actor needs to make a decision to connect/disconnect with other actors in its proximity, some sort of information exchange needs to occur between the two actors to evaluate the utility (happiness) of the undirected tie. Further complexity arises when one of the actors sharing the undirected tie decides to disconnect/remove the undirected tie. Hence, we assumed a directed communication network where each actor only has control over its incoming ties, i.e., each actor identifies the neighbors from which it would like to be receiving information. When an actor requests a directed tie is to be established, the *sender* actor is compelled to form an outgoing tie, i.e., actors do not have any control over their outgoing ties. This does not mean that the actors can change their incoming ties at will, however; the DDIP model controls this process, as will be discussed below.

Let O_i^t denote the set of actor i's out-directed neighbors at period t, i.e., the set of actors to which actor i has an outgoing tie, and let I_i^t denote the set of actor i's in-directed neighbors at period t, i.e., the set of actors from which actor i has an incoming tie.

The variables and parameters listed in Table 1 serve to rigorously present the framework and structure for how the DDIP model forms networks. In each period, all the actors participate in several network building-related activities, performed sequentially in stages: the message forwarding (propagation) stage, the weight and objective calculation stage, and the network state estimation and modification stage, respectively, are described in detail below.

Forwarding stage: message passing procedure

The DDIP model prescribes to begin each period with a forwarding stage. The goal of the forwarding stage is to allow each actor to update its belief about its own position with respect to the rest of the network. This is needed because each actor has only partial information about the global network topology at any point in time: what they

Table 1 Notation, variables, and parameters used in the DDIP model

Variable	Definition
$V = \{1, 2, \ldots, n\}$	Set of actors in the network
$\|V\| = n$	The number of actors in the network
P_i	Set of actors in the proximity of actor *i*
t	Time period
$t_{\max}$	The maximum number of periods to run the DDIP model
O_i^t	The set of actor *i*'s out-directed neighbors at period *t*
I_i^t	The set of actor *i*'s in-directed neighbors at period *t*
g_i^t	The local network structure of actor *i* at period *t*
m	The number of messages each actor sends per period *t*
p_{ijk}^t	The fraction of messages actor *i* receives from source *j* through neighbor *k* during period *t*
α	The probability of a message being terminated
$\beta_{ij}^{\text{own},t}$	Actor *i*'s own (personal) preference for messages from actor *j*, during period *t*
$\beta_{ij}^{\text{neigh},t}$	The average preference of actor *i*'s out-directed neighbors for messages from actor *j* during period *t*
$\beta_{ij}^{\text{total},t}$	The overall preference that actor *i* has for messages from actor *j* at time *t*
ρ	Preference parameter: The weight of an actor's own preferences against it's neighbors preferences
γ, w	Cost parameters: the weights in the objective

know is the number of actors involved in the network formation process, the structure of its local connections with its peers and its proximity restrictions.

The actors engage into the process of passing messages to "learn" which neighbors seem to be close and which neighbors seem to be further away. The messages propagate through the network according do the partially absorbing random-walk procedure; such a random walk has been previously used, e.g., in [26] to define entropy centrality, as well as in [14] to learn and exploring graph structures.

During each forwarding stage, each actor i will send off m messages. Each message is sent with an equal probability to any one of actor i's out-directed neighbors. After being forwarded to a neighbor j, the message could be terminated with a probability α, otherwise it is sent with an equal probability to any one of j's out-directed neighbors. Each message continues to be forwarded between actors until it is terminated. Each message carries with it the information of its source actor (the label or ID of that actor), the last actor who sent it, and a unique identifier that allows any receiving actor i not to double-count the message if its duplicates ever reaches i, which might happen as a result of the message running through a cycle (however, each duplicate will still be forwarded on with a probability α).

Let p_{ijk}^t denote the fraction of messages received by actor i from source actor j, through actor k (meaning k is the actor who had the messages immediately before they reached i), in period t. For application purposes, the message forwarding procedure can be conducted by sending all messages at the same time: i.e., actor i would send an equal fraction of the m messages to each of its out-directed neighbors. For example, if the actors are set to send $m = 1000$ messages per round, an actor with two outgoing ties would send 500 messages to each of these neighbors. Each neighbor that receives the messages then reduces their number by factor α and, in turn, forwards the surviving messages to their out-directed neighbors in even fractions. Once the number of messages received falls under a certain small threshold (e.g., five messages), the propagation is terminated. The p_{ijk}^t value can then be calculated by taking the number of messages that each actor i received from a source actor j through a connected neighbor k and dividing the result by the initial message count m.

Calculation of weights and objective values

In each period t, each actor i uses the values p_{ijk}^t, obtained in the message passing stage in period t, to calculate their objective value: the higher this value, the better network position the actor finds themselves in. In short, the model encourages each actor to be able to receive messages from every other actor in the network. To this end, the actor first calculates their β weights, as described in Table 1.

Let $\beta_{ij}^{\text{own},t}$ represent the level of actor i's own (personal) preference for messages from actor j, during period t, which is computed as

$$\beta_{ij}^{\text{own},t} = e^{(1-(\sum_k p_{ijk}^t))}, \quad \forall j \in V, i \neq j, k \in I_i^t. \tag{1}$$

We model the weight $\beta_{ij}^{\text{own},t}$ as an exponential function to ensure that actor i gives a higher weight to source actors j from whom it does not receive many messages, given the current network structure.

Let $\beta_{ij}^{\text{neigh},t}$ represent the preferences of actor i's out-directed neighbors for messages from actor j during period t. Each actor requests the values of $\beta_{kj}^{\text{own},t}$ and O_k^t from their out-directed neighbors ($k \in O_i^t$), to compute $\beta_{ij}^{\text{neigh},t}$ as

$$\beta_{ij}^{\text{neigh},t} = \frac{1}{|O_i^t|} \sum_{k \in O_i^t} \beta_{kj}^{\text{own},t}, \quad \forall j \in V, i \neq j, k \in O_i^t. \tag{2}$$

The value $\beta_{ij}^{\text{neigh},t}$ is an average of the weights that each out-neighbor of i contains: it reflects a kind of cooperation between actors. The calculation of $\beta_{ij}^{\text{neigh},t}$ allows each actor to make changes that benefit their neighbors as well. Combining (1) and (2), actor i computes their total weights, by averaging these components, weighted with the preference parameter ρ

$$\beta_{ij}^{\text{total},t} = (\rho)\beta_{ij}^{\text{own},t} + (1-\rho)\beta_{ij}^{\text{neigh},t}, \quad \forall j \in V, i \neq j. \tag{3}$$

Note that each actor in the network has only partial information about the network topology at any point in time; let g_i^t be the graph reflecting the structure of its local connections with its peers. Each actor i evaluates their "happiness", given their current local network structure (g_i^t),

$$h_i(g_i^t) = \sum_j \beta_{ij}^{total,t} \sum_k p_{ijk}^t - w \sum_k \gamma |O_k^t|, \quad \forall j \in V, i \neq j, k \in I_i^t. \tag{4}$$

For clarity, note that the terms "happiness value" and "objective value" are henceforth used interchangeably. Equation (3) captures how "happy" actor i is about their current local network structure. The larger the fraction of the messages that actor i receives from all other actors in the network, the larger the objective value.

To discourage overloading an actor with forwarding too many messages, there is a cost that is directly related to the number of the outgoing ties, O_k^t, that each in-neighbor k of actor i possesses in time period t. This cost is expressed using parameters γ and w, intuitively, the parameter γ captures the neighbor's cost for a connection while the parameter w captures the scaling parameter for the cost and makes up the right portion of the objective equation, as reflected in (4). The cost parameter ensures the actors from having too many outgoing ties, without placing any absolute restrictions on the number of ties an actor can possess. The use of both parameters allows for tuning the different costs used in the model; giving more flexibility to these parameters allows for a model that can create more robust networks. As actor i adds an incoming tie to its neighbor k who possesses more outgoing ties, actor i's own cost will increase, which will decrease actor i's own objective value.

As a final remark, note that the β weights are re-evaluated in each period and are used to (re-)compute the objective function as the global network structure changes over time.

Network estimation stage and modification stage

After each actor has computed their own objective value, they have to make the decisions on whether or not to make modifications to their local networks. This section describes how potential modifications are evaluated and compared, and how actors decide which ones to make.

In each period, each actor has three different options: they can add an incoming tie to one of their eligible neighbors, delete a currently active incoming tie from one of their connected neighbors, or do nothing (an actor cannot simultaneously add and delete a tie in the same period). Each actor estimates how their objective might change from making a particular modification, with the help of the message passing process observations collected in prior stages; each actor shares their p_{ijk}^t values and their outgoing ties information O_i^t with the unconnected neighbors in their proximity.

An actor i estimates the value gain that will result from adding an incoming tie to an eligible neighbor j ($i \leftarrow j$) by computing

$$h_i(g_{i\leftarrow j}^t) = h_i(g_i^t) + \frac{1}{|O_j^t|+1}\sum_{u}\sum_{k\,\in\, I_j^t} p_{juk}^t - w\gamma(|O_j^t|+1), \quad \forall k \in I_j^t, u \in V \setminus \{i,j\}, j \in P_i \backslash I_i^t. \tag{5}$$

Here, actor i looks at the current objective value and adds the estimated increase due to the fraction of the messages that i would receive from each other actor in the graph if it were to connect to actor j—this is captured in the middle portion of (5). Then, the actor subtracts the cost of pulling messages from actor j. Similarly, the actor estimates the potential effect of disconnecting from one of their currently connected in-neighbors, j ($i \not\leftarrow j$), by computing

$$h_i(g_{i\not\leftarrow j}^t) = h_i(g_i^t) - \sum_{u}\sum_{j\,\in\, I_i^t} p_{iuj}^t + w\gamma|O_j^t|, \quad \forall\, u \in V\backslash\{i,j\}, j \in I_i^t. \tag{6}$$

Here, the actor takes the current objective value and subtracts the fraction of messages that were received from all actors $u \in V \setminus \{i, j\}$ that came through actor j; then, the cost of pulling from actor j is added (this cost would no longer be incurred). Note that the logic of (6) is not flawless, as actor i might be receiving a large fraction of the same messages, which came through j, from another actor, so i would believe that losing j would be costly while it might not be the case.

The DDIP model imposes rules that actors follow when deciding which modifications should be made to their local network. The actors that have any incoming ties under a certain limit ($l = 2$) are considered isolated, and these actors are not allowed to lose out-directed ties to discourage complete isolation of actors. Having obtained the estimates per (5) and (6), each actor will make the modification that leads to the largest increase in their objective value. Once all actors have made their choices, the next period starts.

The presented description of the DDIP model logic motivates us to understand the impact of model parameters on the DDIP model outcomes. Indeed, a low value of γw ($0 < \gamma w < 0.2$) is likely to lead to networks which are fully connected, as the cost associated with adding a tie is almost negligible. Meanwhile, a high value of γw ($10 < \gamma w < \infty$) is likely to lead to networks which are disconnected, as the cost associated with adding a tie is not worth the messages received as a result of adding the tie. Hence, a parameter search optimization problem is to determine a well-balanced objective function.

Parameter estimation

It is paramount to find the input parameters for the DDIP model which would allow for the formation of networks that are robust and efficient for information propagation. In this regard, we formulate an optimization problem to search through the parameter space efficiently, and utilize a modified Nelder–Mead algorithm to find the most effective parameters for a variety of graph sizes.

Due to the iterative nature of the Nelder–Mead algorithm, the DDIP model would have to be evaluated many times for each parameter setting and graph size. Further, for each run, the DDIP model would have to simulate the entire path of each message passed in each period, which proves to be too computationally intensive.

To meet this challenge, in this section, we make advances to employ Markov Chain theory to replace the *message-passing* component in the DDIP simulation runs by more accurate and much faster analytical analyses. Specifically, a Markov Chain approximation for the message forwarding stage is designed, allowing for calculations of expected fraction of messages received between all actors in the network to be computed.

The DDIP model introduced in the paper implements a decentralized approach for the network formation and works on partial information, however, to evaluate the networks generated, we need the complete network topology. In this regard, to evaluate the networks generated while performing the parameter search, we consider an oracle with access to complete information, i.e., complete network topology. We utilize the oracle to quickly compute certain network-related metrics, which enables the efficient parameter search.

The flow of this section is as follows: first, introduce the Markov Chain approximation for the message forwarding stage; second, to quantify the topological properties of the networks formed under the DDIP model, we introduce the evaluation metrics. Third, we formulate the parameter search optimization problem.

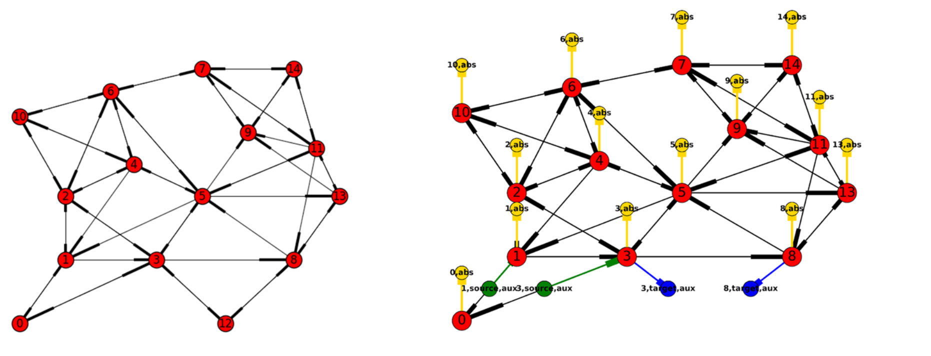
Markov chain approximation for the message forwarding stage

The Markov Chain approximation for the message forwarding stage allows for calculating the expected fraction of messages received between all actors in the network, while not having to simulate the path of every message taken through the network to termination. The approximation drastically reduces the amount of computation necessary to parameterize the DDIP model.

To calculate the expected fraction of messages any actor i receives from source actor j, through in-directed neighbor k, we build an auxiliary Markov Chain and compute its absorption probabilities. We define a separate Markov Chain for each pair "source actor j–target actor i", where auxiliary states are defined to emulate the message passing procedure.

Specifically, each unique Markov Chain has multiple states introduced in conjunction with each actor: an *absorbing auxiliary state* for each actor to represent the termination of the message passing process, an *auxiliary source state* for each out-directed neighbor of source actor j, and an *auxiliary target state* between the target actor i and each of its in-directed neighbors k (see Fig. 1). The auxiliary source states are used to ensure messages do not get terminated when they are first sent, while the auxiliary target states are absorbing and are created to calculate the probability that a message from source j will reach target i through a specific in-directed neighbor k.

The absorbing random walk process that specifies the propagation of messages in each period through the network that is being built, given the DDIP model parameters, can be analyzed to compute the transition probabilities for the above-described Markov Chains, to be used to speed up the simulation of the message-passing. In doing so, we calculate the probability of a message being absorbed into an auxiliary target actor, to get the p^t_{ijk}. Using the transient sub-matrix Q of the transition probability matrix, the identity matrix I (of the same size), and the target/in-directed neighbor column of the absorption sub-matrix R (denoted $R_{i,k}$) for target i and neighbor k, one has



a Original Sample Graph **b** Markov State Diagram for the sample Graph

Fig. 1 **a** A sample graph created with 15 actors to depict a sample network. **b** The Markov State diagram with actor 0 as source and actor 12 as target; the auxiliary source states are shown in green, the absorbing auxiliary termination states are shown in yellow, and target auxiliary states which are also absorbing are shown in blue

$$p_{ijk}^{t} = (I - Q)^{-1} R_{i,k}\,, \quad \forall\, i,j \in V, i \neq j, k \in I_i^t. \tag{7}$$

The result in (7) is proven in [27] using a slightly different notation. Applying (7) for each combination of source j, target i, and in-directed neighbor k of i, the values p_{ijk}^t can be calculated for each period t exactly.

The p_{ijk}^t values were originally computed by simulating the path of all m messages generated per actor until termination. This original simulation would require $\mathcal{O}(m * n * (1/\alpha))$ calculations, whereas the presented Markov Chain approximation requires about $\mathcal{O}(n^2)$ calculations to obtain the same values; besides, it does so exactly, unlike a set of simulation runs. This shows a drastic decrease in the amount of computation needed to obtain p_{ijk}^t, since m is much larger than n.

The presented Markov Chain approximation procedure opens the door to the development of iterative model parameterization algorithms that would run in reasonable time. In this paper, the Nelder–Mead algorithm will be employed to this end. The following subsections describe how this is accomplished.

Evaluation metrics

To evaluate the quality of the networks generated using the DDIP model, we consider an all-knowing oracle which has the complete network information during any time period. We utilize the oracle to compute certain network-related metrics which aid in the parameter search.

Let $G = (V, S)$ denote the fully connected network, with the nodes $V = \{1, 2, \ldots, n\}$ and edge set $S \subset V \times V$, consistent with the proximity restrictions P_i. One could also consider $G = (V, S)$ as the input network for the DDIP model, where S reflects the proximity constraints P_i. Let $g^t(V, E^t)$ denote a DDIP model-generated network at time period t.

We introduce three metrics for examining the quality of a given solution: the edge ratio (e_r), the average inverse distance between each pair of actors in the graph (μ), and the edge efficiency (η).

The edge ratio, e_r,

$$e_r = \frac{|E^{t_{\max}}|}{|S|}, \tag{8}$$

is used to describe the ratio of the number of active edges $|E|$ used in an output graph from the DDIP model to the number of possible edges $|S|$. A small e_r means that the resulting network of the DDIP model contained relatively few edges compared to the total number possible.

The average inverse distance (a.k.a network efficiency [28]), μ, is equal to the average of the sum of the inverse lengths of the shortest paths π_{ij} between all pairs of actors i and j,

$$\mu = \frac{1}{n(n-1)} \sum_{i} \sum_{j \neq i} \frac{1}{\pi_{(i,j)}}. \tag{9}$$

As μ increases, the average number of edges it takes to go from one actor to any other actor decreases. This statistic is useful for taking into account graphs that are not fully connected, since the inverse distance between a pair of unconnected actors is 0 $(1/\infty)$.

The edge efficiency, η, combines the information captured in both e_r and μ to evaluate the quality of solutions produced by the DDIP model

$$\eta = \frac{e^{a\mu}}{e_r}\frac{n}{|S|} = \frac{ne^{a\mu}}{|E^{t_{\max}}|}. \tag{10}$$

To place emphasis on μ in η, we scale the statistic by an exponential function $e^{a\mu}$, where a can be altered to decide how much to weigh μ. The η value is also weighted by $|s|$/n to take into account the natural density of the sample input graph, allowing to compare less sparse graphs to dense graphs with ease.

When evaluating solutions (of the parameter search), the produced graphs that have a higher value of η are preferred, as this favors graphs that achieve a smaller average distance between actors, while using relatively few edges. We now have the necessary framework to formulate a parameter search optimization problem using the evaluation metrics (8–10).

Parameter search optimization problem formulation

We formulate a parameter search optimization problem with the goal of finding the DDIP model parameters that would result in the DDIP model producing networks which are efficient and robust with respect to the propagation of information.

There are two main input parameters that are needed to run the model: (1) the cost of connecting to a neighbor, per each of a neighbors out-directed ties γ, and (2) the scaling factor for neighbor connection cost w. To search for values of these parameters that would produce the best results, the following parameter search optimization problem is formulated:

$$\underset{\gamma,w}{\text{Maximize}} \quad f(x(\gamma,w)) = \frac{1}{|\Omega_n|}\sum_{S\,\in\,\Omega_n} \eta_S^{x(\gamma,w)}. \tag{11}$$

The maximization problem in (11) is that of finding the set of parameters w and γ that will maximize the average resulting η from running the DDIP model over set of sample input graphs Ω_n of the same size n, while varying the possible edge set S, where x is the parameter settings used in the DDIP model for γ and w. We consider the multiple input graphs with varying edge sets in order for the model being applicable to any input graph of size n; in this regard, multiple synthetic graphs were created upon which the parameter search was run, as detailed in the next section.

Nelder–Mead algorithm

The Nelder–Mead algorithm was first developed in [29], with the goal of defining an algorithm that would be able to search for a good sets of vertices to sample when minimizing an objective equation.

The Nelder–Mead simplex search algorithm is initialized by an input set of vertices of size "n" (different than n, the number of actors in the graph G), decided by the user, which in this case is an initial set of scaling parameters w_i, $i = 1, 2, \ldots, \mathrm{n}+1$ and γ_i, $i = 1, 2, \ldots, \mathrm{n}+1$. Recall that the form of parameter set (pairs) to be searched over is denoted by $x(\gamma_i, w_i)$ for $i = 1, 2, \ldots, \mathrm{n}+1$. After the initial vertices are inputted into the algorithm, the parameters are evaluated using the DDIP model, and re-assigned indexes in order of decreasing η^{x_i}. Specifically, $x_1(\gamma_1, w_1)$ represents the maximum value of η^{x_i}, whereas x_{n+1} represents the minimum value of η^{x_i} in the simplex (the list of parameters $x_i, i = 1, 2, \ldots, n+1$).

Using the sorted simplex and the centroid of parameters x_i, $i = 1, 2, \ldots, n$, each iteration of the Nelder–Mead algorithm will search for new vertices to introduce into the simplex: reflection points, expansion points, and contraction points. These points look to introduce new vertices into the simplex that will allow the algorithm to find new maximums. If none of the points introduced are evaluated to be larger than the worst point in the simplex x_{n+1}, the algorithm will shrink the simplex towards the best parameter x_1. The algorithm continues to evaluate and introduce points into the simplex until a pre-set stopping criterion is met and returns the best parameter set found.

The pseudocode of the DDIP-specific variation of the Nelder–Mead algorithm to solve (11) can be found in Appendix (see Algorithm 3).

Computational experiments

To test if the input parameters to the DDIP model, returned by our parameter estimation algorithm, are suitable for building a desirable connection structure for any network of size n, we create multiple synthetic networks, of a form typically found in UAV-related applications, upon which we first perform the parameter estimation. This section describes the creation of the synthetic networks. Further, it explains the experimental setup under which we perform the parameter estimation through the implementation of Nelder–Mead algorithm. Finally, the results of this venture are assessed in the testing and evaluation stage, where the proposed optimized DDIP model outcomes are systematically compared against the centralized network formation approach, which employs Mixed Integer Programs that provide optimal communication network solutions given the original layout, i.e., feasible communication channels, of any network.

Synthetic network generation

Towards the synthetic generation of various topologies for the node set of the same size n, two methods were used in this paper: a unit disk method and Exponential Random Graph Model (ERGM)-based method. The workings of these generators are described in the following subsections.

Unit disk method

The method is fairly simple, yet follows the communication restrictions for the model closely. The unit disk method places n actors randomly on an (x, y) plane: the positions of the actors, x and y, are uniformly distributed between [0, 100]. The allowable edges of the graph are decided by a distance parameter d. An edge is allowed between two actors if the Euclidean distance between the actors is less than d. This follows the assumption that actors can only connect with each other directly if they are in close enough proximity of each other. The d parameter for the method was increased until it was just large enough that each actor had at least two connections available, without loss of generality. To illustrate the networks generated using the unit disk method, see Fig. 2.

ERGM method

The use of ERGM or p^* graphs is widespread in the domain of social network analysis. ERGM network generation was studied extensively in [30], where Snijders describes in detail how ERGM parameters are fit to data (an observed network sample) using statistical analysis and Markov Chain Monte Carlo estimation techniques. Due to the ability of the ERGM model to generate networks with similar structural properties to those of an observed sample, we are able to generate graphs that resemble UAV networks, in a systematic manner. The ERGM graphs were generated by performing MLE on the structural properties of sample UAV networks given, for varied network sizes.

To create the ERGM models for each graph size n, sample "observed" graphs were loaded into "R" software, which was used to fit, test, and simulate the ERGM graphs. The ERGM packages in "R" work with a predefined set of "structural signatures" or "statistics" (such as the number of edges, number of actors with certain degrees, geometrically weighted edge shared partners, etc.) and quantify how important each of those is to the overall structure of the graph, based on how often the statistic appears in the observed graph. To discourage disconnected sample graphs, the ERGM model for most graph sizes would include the isolates statistic. Since each model was fitted to graphs without any isolates, this statistic value is set to $-\infty$, disallowing isolates in simulated graphs.

If specified, ERGM models will also look at the prevalence of attributes that the observed graph contains. To try to control for the geometric distance assumption, a "closeness" parameter was assigned to each actor in the observed graph. This parameter was set by assessing how close each actor was to actor 1, meaning that actors that were closer to 1 would be given the value of 2 or 3, whereas actors that were further away would be given larger values. This attribute allowed for the ERGM model to gauge how geographically close actors would affect their proclivity for connecting. Using the "smalldiff" statistic, the absolute difference between each pair of actors' closeness values was measured; the model then was used to produce graphs where only adjacent actors

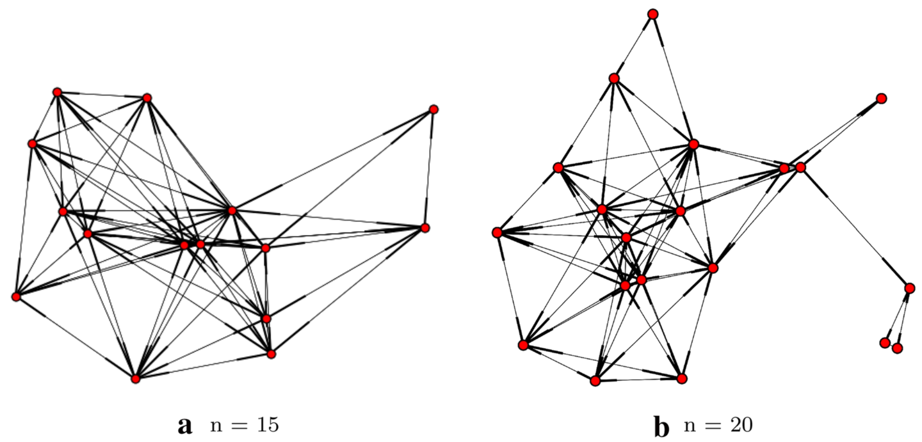


Fig. 2 Synthetic sample graphs generated from Unit Disk Method

would be able to connect to each other. This becomes possible because the "smalldiff" parameter makes actors with smaller differences in the closeness values more likely to be potentially connected.

After the parameters were fitted to the observed graphs, a brief goodness of fit test was performed to make sure the output graphs were similar to the observed input. Graphs were then simulated and captured to be used as sample instances to run the DDIP model on. Since the ERGM parameters for each graph size are not scalable, then a different set of statistics was obtained for varied network sizes (see Appendix for details).

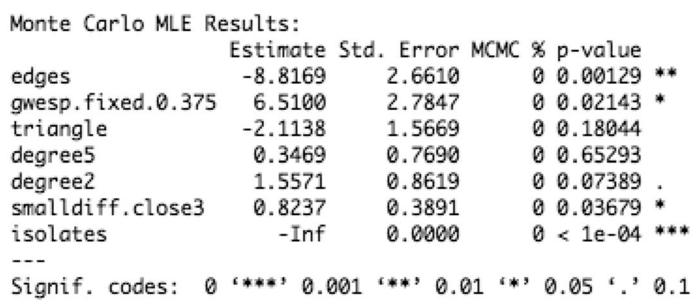
To illustrate the significant statistics in an ERGM model, we fit an observed UAV network of size $n = 15$ (refer to Fig. 3). Further, we generated the synthetic networks with the significant statistics of an observed sample. Figure 4 illustrates how the input ("observed") UAV network compares to a representative synthetically generated ERGM network.

From the utilization of the networks that were generated from the unit disk and ERGM methods, the DDIP now had a suitable test bed of sample input graphs to run the parameter search optimization problem to infer the DDIP model input parameters.

Experiment setup

To ensure that the developed Nelder–Mead algorithm works with input graphs Ω_n that are diverse, half of the sample graphs in Ω_n were generated using the Unit Disk model, and the other half were generated using the ERGM model. The sample input set Ω_n was built so as to make sure that the Nelder–Mead algorithm searches for parameters that maximize η^{x_i} for all graphs of size n, so that the effects of structural differences between sample graphs in Ω_n would be mitigated. To reduce the computational time, the algorithm termination parameter was set for the number of iterations without output improvement. If the algorithm would not find a new maximum after a pre-specified number of iterations, it would terminate.

Recall that the Nelder–Mead algorithm is used to identify the best DDIP parameters which allow for the formation of networks that are robust and efficient for information exchange. Upon identifying the best DDIP parameters, for a network of specific size,



```
Monte Carlo MLE Results:
                   Estimate Std. Error MCMC % p-value
edges               -8.8169     2.6610      0 0.00129 **
gwesp.fixed.0.375    6.5100     2.7847      0 0.02143 *
triangle            -2.1138     1.5669      0 0.18044
degree5              0.3469     0.7690      0 0.65293
degree2              1.5571     0.8619      0 0.07389 .
smalldiff.close3     0.8237     0.3891      0 0.03679 *
isolates               -Inf     0.0000      0 < 1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fig. 3 ERGM fit to an observed UAV network with 15 actors to identify the significant statistics. Note that the following statistics are found significant in this fitting exercise: the number of edges in the graph, the geometrically weighed edgewise shared partner term (gwesp), the small difference attribute, and the number of isolates. Based on the computed parameter estimates, this model is likely to produce graphs with relatively few edges and shared connections between neighbors; also, the directly connected neighbors in the resulting networks would tend to see small differences in closeness values

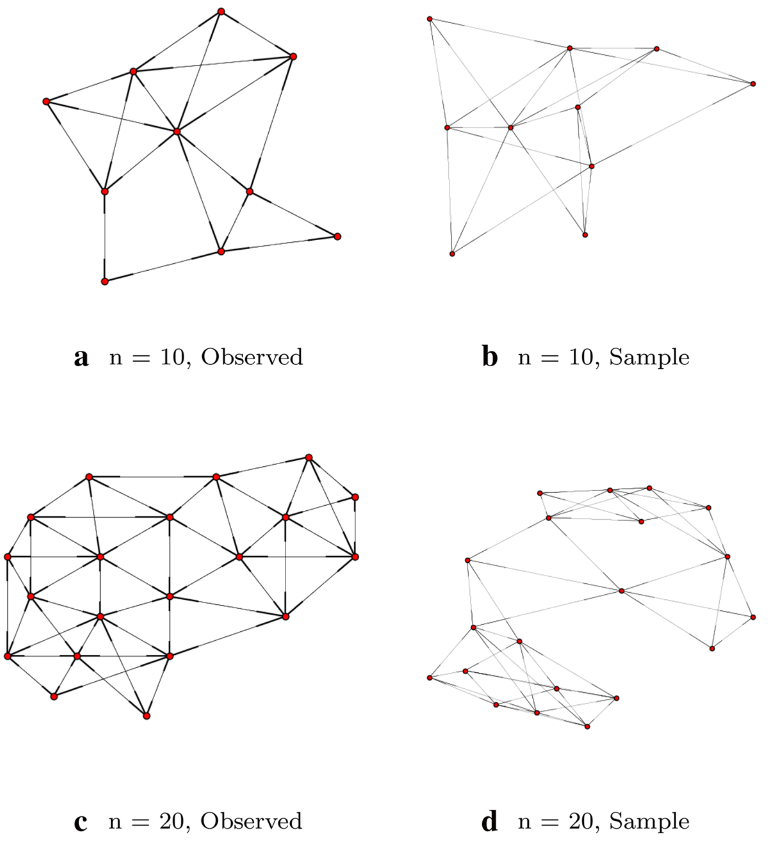


Fig. 4 Comparison of synthetic ERGM-generated sample networks with the observed UAV network. **a**, **b** synthetic sample networks vs. the input ("observed") network with 10 actors; since the size of graph was relatively small, the simulated sample graphs tended to be very similar in shape. **c**, **d** synthetic sample networks vs. the input network with 20 actors; sample networks did not have ties reaching across the entire graphs, meeting the requirement that only close actors should be able to connect

from the Nelder–Mead algorithm, the actors can implement the DDIP to form a robust communication network. In this regard, we can implement the parameter estimation in a centralized manner, i.e., offline in a computer and the identified best parameter values can be provided as an input to actors for implementing DDIP. The parameter estimation needs to be performed only once, for each network size. Hence, even under limited computational capabilities, the actors can implement DDIP.

Nelder–Mead implementation

The Nelder–Mead algorithm was implemented for graphs of size $n = 10$, 15, and 20. For each graph size, the scaling weight a for average inverse distance in the edged efficiency in (10) was set to 3.15. This value was found fitting to produce edge efficiency values that were more balanced between the inverse distance of the outputs and the number of edges used in the graph. Each algorithm run was initialized with a set of six pre-defined parameter vertices that varied slightly between graph sizes. These initial vertices were set based on initial testing of the DDIP model and the expected calculations for the local objective function values for actors in the graph.

For graphs of size $n = 10$ and 15, ten input sample graphs were evaluated: five Unit Disk graphs and five ERGM graphs. For these graph sizes, the maximum number of stalling iterations was set to five as the termination parameter. For the graphs of size $n = 20$, six sample input graphs were evaluated. Also, the maximum number of stalling iterations was set to three as the termination parameter. Further, each DDIP model run was initialized with each actor connecting to one randomly chosen neighbor, to see how the DDIP model solutions vary with different initial connection setups.

The results of the Nelder–Mead algorithm runs are shown in Table 2. The algorithm preferred a larger γ value for $n = 20$, and smaller γ values for $n = 10$ and 15. Also, the algorithm preferred a smaller w value for $n = 20$. Overall, the values for γ and w were a bit larger than we expected, which caused the e_r value to be fairly small, while somewhat sacrificing the average optimal μ for each graph size. Also, the scaling factor a in η was found to play a big role in how the model reacts.

To compare the DDIP model generated networks with the fully connected network would be naive. So, we decided to compare the DDIP model-generated networks with the networks generated in a centralized setting. We approached this challenge by building a Mixed Integer Program as described in the next section.

Optimal network configurations

To find the optimal network configurations, given a set of allowable edges S, a limit on the number of active connections $|E|$, and a maximum possible graph efficiency μ, we use the linear mixed integer programming (MIP) formulation technique which is similar in spirit to the one developed in [31]. The key difference is that in this paper, the network needs to be directed and another cohesiveness measure is considered. Our goal is to find optimal network configuration given the limit on the number of connections, rather than to minimize the number of connections given the cohesiveness requirement. Nevertheless, the technique that allows us to model distances in a graph using linear constraints is similar. For further reference, see a solution to a related problem in [32].

Let binary variables x_{ij} for all allowable edges $(i,j) \in S$ define the set of active edges E, i.e., $x_{ij} = 1$ if $(i,j) \in E$ and, $x_{ij} = 0$ if allowable edge is not active. To model the distances in the network constructed by active edges, the MIP formulation uses the following extra variables:

- $w_{ij}^{(\ell)}$ for all $i,j = 1,\ldots,n;\ i \neq j;\ \ell = 1,\ldots,L$: $w_{ij}^{(\ell)} = 1$ if there is a path of length at most ℓ from node i to j in a subgraph defined by active edges (s, t) with $x_{st} = 1$, and 0 otherwise;

Table 2 Nelder–Mead algorithm parameter results

Graph Size	γ	w
$n = 10$	0.642	1.096
$n = 15$	0.555	1.210
$n = 20$	1.288	0.863

- $y_{ikj}^{(\ell)}$ for all $i,j = 1,\dots,n;\ i \neq j \neq k;\ \ell = 2,\dots,L, (i,k) \in S$: $y_{ikj}^{(\ell)} = 1$ if there is a path of length at most ℓ from node i to j in a subgraph defined by active edges (s,t) with $x_{st} = 1$ which goes through a node k, and 0 otherwise.

Note that the distance variables are defined for $\ell = 1,\dots,L$, where L is some constant whose appropriate choice will be described later. Observe that if $w_{ij}^{(\ell)} - w_{ij}^{(\ell-1)} = 1$ for a some ℓ and a pair of nodes i, j, then the distance from node i to j is equal to ℓ. Therefore, the number of pairs of nodes with distance (shortest path length) exactly ℓ ($\ell = 2,\dots,L$) is equal to

$$\sum_{i,j=1:i\neq j}^{n} (w_{ij}^{(\ell)} - w_{ij}^{(\ell-1)}),$$

and the efficiency μ of the graph of active connections E can be written as

$$\mu = \frac{1}{n(n-1)} \sum_{i,j=1:i\neq j}^{n} \left(w_{ij}^{(1)} + \sum_{\ell=2}^{L} \frac{1}{\ell} \left(w_{ij}^{(\ell)} - w_{ij}^{(\ell-1)} \right) \right), \tag{12}$$

if L is greater than the maximum possible finite distance (diameter) in the graph of active connections. Then, the linear problem formulation which minimizes μ given the limit on the number of active connections K can be written as follows:

Problem 1 (MIP for optimal network configuration)

$$\max\ \mu = \frac{1}{n(n-1)} \sum_{i,j=1:i\neq j}^{n} \left(w_{ij}^{(1)} + \sum_{\ell=2}^{L} \frac{1}{\ell} \left(w_{ij}^{(\ell)} - w_{ij}^{(\ell-1)} \right) \right) \tag{13a}$$

subject to

$$w_{ij}^{(1)} = x_{ij}, \quad \forall (i,j) \in S \tag{13b}$$

$$w_{ij}^{(1)} = 0, \quad \forall (i,j) \notin S, i \neq j \tag{13c}$$

$$w_{ij}^{(\ell)} \geq w_{ij}^{(\ell-1)}, \quad \forall i,j \in V, i \neq j,\ \ell \in \{2,\dots,L\} \tag{13d}$$

$$w_{ij}^{(\ell)} \leq x_{ij} + \sum_{k\neq j:(i,k)\in S} y_{ikj}^{(\ell)}, \quad \forall (i,j) \in S,\ \ell \in \{2,\dots,L\} \tag{13e}$$

$$w_{ij}^{(\ell)} \leq \sum_{k\neq j:(i,k)\in S} y_{ikj}^{(\ell)}, \quad \forall (i,j) \notin S, i \neq j,\ \ell \in \{2,\dots,L\} \tag{13f}$$

$$y_{ikj}^{(\ell)} \leq x_{ik},\ y_{ikj}^{(\ell)} \leq w_{kj}^{(\ell-1)}, \quad \forall i,j \in V, (i,k) \in S, i \neq j \neq k, \ell \in \{2,\dots,L\} \tag{13g}$$

$$\sum_{(i,j)\in S} x_{ij} \le K, \tag{13h}$$

$$x_{ij}, \in \{0,1\}, \quad \forall (i,j) \in S \tag{13i}$$

$$w_{ij}^{(\ell)},\, y_{ikj}^{(\ell)} \in \{0,1\}, \quad \forall i,j \in V,\ (i,k) \in S, i \neq j \neq k, \ell \in \{1,\dots,L\}. \tag{13j}$$

Observe that the problem formulation models distance variables $w_{ij}^{(\ell)}$ and $y_{ikj}^{(\ell)}$ recursively in such a way that if there is no path of distance at most ℓ from node i to node j which uses the set of active connections (s,t) such that $x_{st} = 1$ then $w_{ij}^{(\ell)} = 0$ and $y_{ikj}^{(\ell)} = 0$ (constraints 13b–13g). If the distance from node i to j is $\ell > 1$, it might be feasible to have $w_{ij}^{(\ell)} = 0$ and $y_{ikj}^{(\ell)} = 0$ as well. However, this cannot happen in an optimal solution, and to demonstrate that, we note that for a given pair of nodes i, j its contribution to the objective function (13a) without scaling can be rewritten as

$$\begin{aligned} w_{ij}^{(1)} + \sum_{\ell=2}^{L} \frac{1}{\ell}\left(w_{ij}^{(\ell)} - w_{ij}^{(\ell-1)}\right) &= w_{ij}^{(1)} + \frac{1}{2}\left(w_{ij}^{(2)} - w_{ij}^{(1)}\right) + \cdots + \frac{1}{L}\left(w_{ij}^{(L)} - w_{ij}^{(L-1)}\right) \\ = \left(1 - \frac{1}{2}\right)w_{ij}^{(1)} &+ \left(\frac{1}{2} - \frac{1}{3}\right)w_{ij}^{(2)} + \cdots + \left(\frac{1}{L-1} - \frac{1}{L}\right)w_{ij}^{(L-1)} + \frac{1}{L}w_{ij}^{(L)}. \end{aligned} \tag{14}$$

Observe that each term $w_{ij}^{(\ell)}$ for any $\ell = 1,\dots,L$ comes with positive coefficient, which, taking into account the maximization nature of the problem, guarantees that in an optimal solution $w_{ij}^{(\ell)} = 1$ for all $\ell \ge \pi_{ij}$ (the length of the shortest path from node i to j).

To ensure that the objective function of the optimal solution of the formulation above corresponds to the correct value of graph efficiency (μ) and the obtained solution is indeed the graph with the maximum possible efficiency μ, one can use $L = n - 1$. However, using large values of L results in larger MIP formulations since the number of variables and constraints grows linearly with L.

Hence, to use as smallest L as possible, we employ sequential solution technique (Exact Iterative Algorithm described below), which is also similar to the one described in [31, 32]. Specifically, we set $L = \mathrm{diam}(G)$ (graph diameter, the largest finite distance in a graph) first and solve this problem, where $G = (V, S)$ consists a set of all allowable connections. If the diameter of the obtained network is greater than L, then we increase L and solve the problem again until the diameter of the obtained network will not be greater than L. This technique allows to substantially reduce the computational time and the amount of variables. The next proposition provides a formal proof that at the end such a procedure returns an optimal network configuration. Let $diam(G)$ and d_{ij} denote the diameter of a graph G and a distance from node i to node j (length of the largest shortest path), and let $\mathbb{1}_{d_{ij}\le L}$ be the indicator function which is equal to 1 if $d_{ij} \le L$ and 0, otherwise. Then, the following proposition holds.

Proposition 1 *Let E^* be the optimal solution of Problem 1 for some $L = L_0$ and $G^* = (V, E^*)$. If $\mathrm{diam}(G^*) \le L_0$, then G^* is also an optimal solution of Problem 1 for $L = n - 1$.*

Proof For any graph $G = (V, E)$ let $\lfloor \mu \rfloor_L = \frac{1}{n(n-1)} \sum_{i,j=1:i\neq j}^{n} \frac{1}{d_{ij}} \mathbb{1}_{d_{ij}\leq L}$, which can be viewed as a truncated version of the graph efficiency μ, where any distance of greater than L in a graph G is treated as if the nodes are disconnected ($d_{ij} = \infty$). In fact, the objective function of Problem 1 for any given L represents the truncated version of a graph efficiency $\lfloor \mu \rfloor_L$.

Note that any feasible solution of Problem 1 with $L = n - 1$ is also a feasible solution for Problem 1 for any given $L = 1, \ldots, n - 1$. Conversely, from any optimal solution of Problem 1 for any given $L = 1, \ldots, n - 1$ the feasible solution of Problem 1 can also be easily constructed using the recursive procedure. Moreover, note that $\lfloor \mu \rfloor_L \leq \mu$ for any $L = 1, \ldots, n - 1$, and , if $L \geq \text{diam}(G)$, then $\lfloor \mu \rfloor_L = \mu$, which ends the proof of the proposition. □

Algorithm 1 gives the formal description of the steps that we use to find the optimal network configurations in accordance with our MIP.

Algorithm 1 Exact Iterative Algorithm

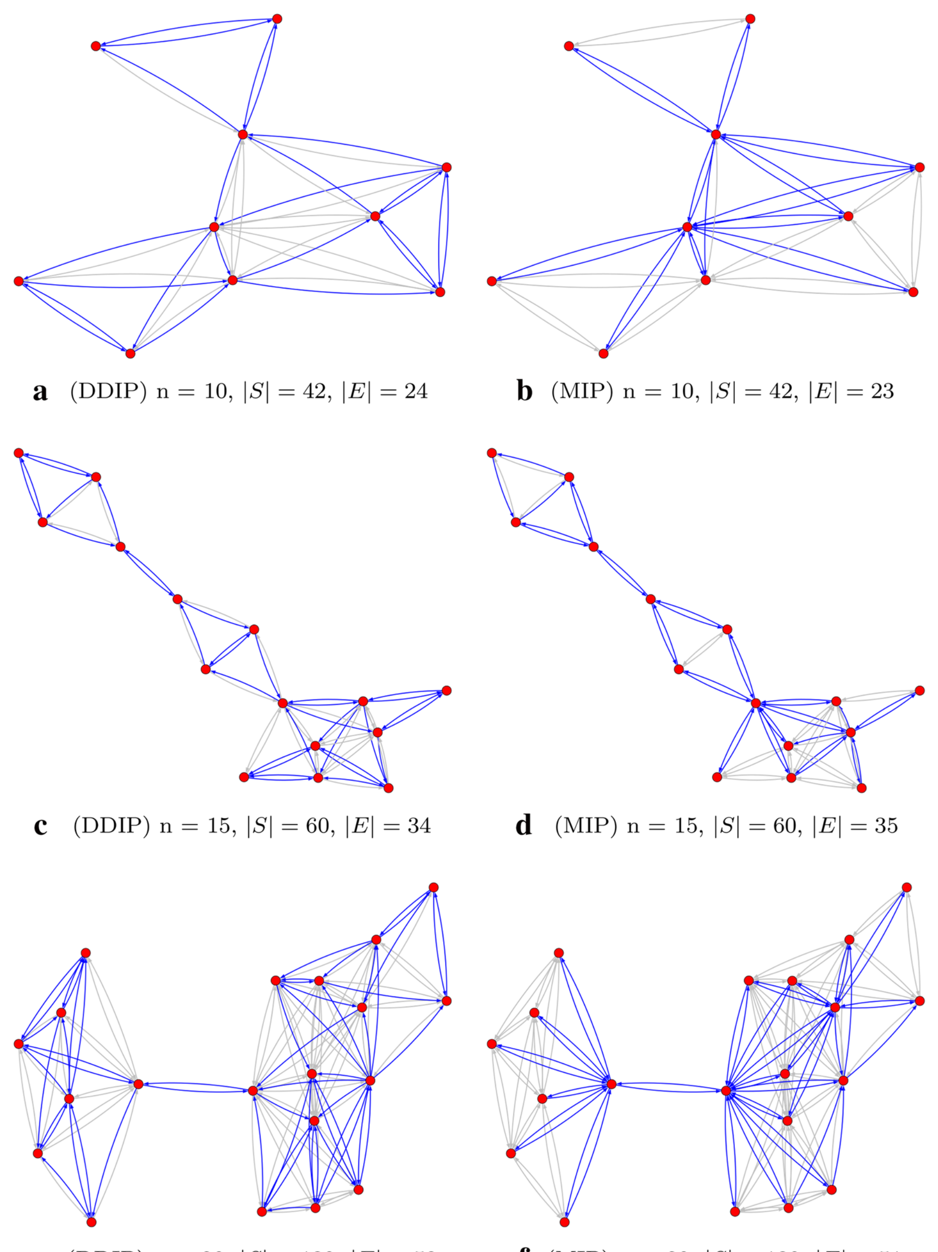
```
    Input: A graph $G = (V, S)$ and a limit on the number of active connections $K$
    Output: Subset $E^* \subseteq S$
 1: begin
 2:   $L_0 \longleftarrow diam(G)$
 3:   $E^* \longleftarrow$ optimal solution of Problem 1 for $L = L_0$
 4:   $G^* \longleftarrow (V, E^*)$
 5:   $L_1 \longleftarrow diam(G^*)$
 6:   while $L_1 > L_0$ do
 7:     $L_0 \longleftarrow L_1$
 8:     $E^* \longleftarrow$ optimal solution of Problem 1 for $L = L_0$
 9:     $G^* \longleftarrow (V, E^*)$
10:     $L_1 \longleftarrow diam(G^*)$
11:   end while
12:   return $E^*$
13: end
```

Experimental results

Multiple combinations of the DDIP model parameters were tested and best selections identified. For the test instances with the networks of size $n = 10, 15, 20$, the w parameters used were 0.642, 0.555, and 1.288, and the γ parameters used were 1.096, 1.210, and 0.863, respectively.

The DDIP instances were tested using Python 2.7 on a Dell Inspiron 15 7000 (2.3 GHz Intel Core i5-6200U processor, 8 GB RAM) for ten periods, ten times per test input graph, per each graph size. The MIP formulations were implemented and solved with Gurobi Optimizer 8.1 [33] using Python interface. See Figs. 5 and 6 for the comparison between the networks generated with the DDIP model and the networks generated by the MIP (Problem 1) for each test instance.

The values of the network performance metrics were calculated for all the resulting networks that were generated with the DDIP model. The values of μ and η were also computed for the fully connected instances ($e_r = 1$), and also, for the MIP solutions (obtained under the constraint $|E|^{\text{MIP}} = Avg|E|^{\text{DDIP}}$), for each sample graph. Table 3 reports the average results for each performance metric and compares the results of the DDIP model to the metrics of the fully connected graphs and MIP solutions for each graph size and type.



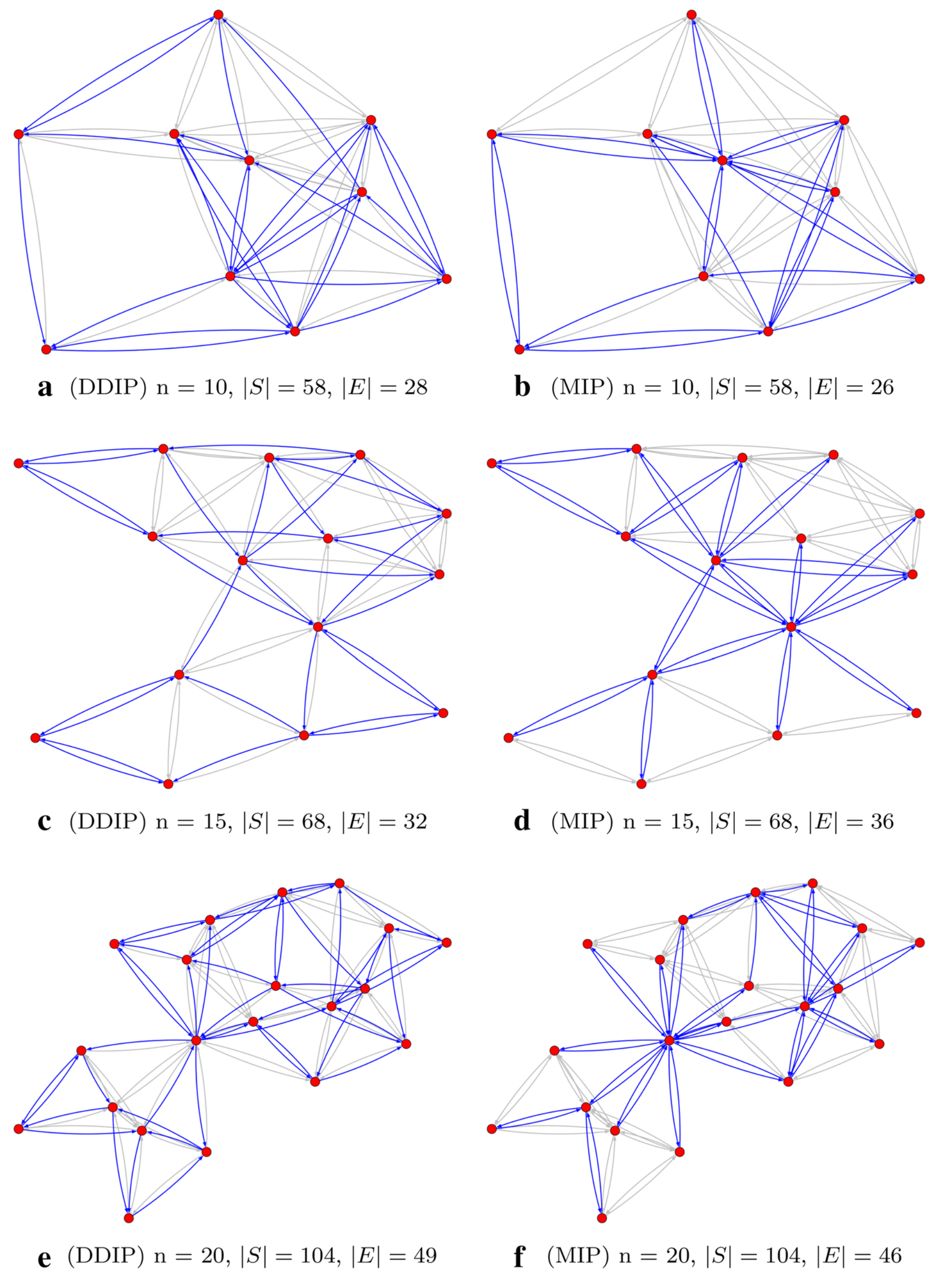
a (DDIP) $n = 10,\ |S| = 42,\ |E| = 24$ **b** (MIP) $n = 10,\ |S| = 42,\ |E| = 23$

c (DDIP) $n = 15,\ |S| = 60,\ |E| = 34$ **d** (MIP) $n = 15,\ |S| = 60,\ |E| = 35$

e (DDIP) $n = 20,\ |S| = 130,\ |E| = 58$ **f** (MIP) $n = 20,\ |S| = 130,\ |E| = 51$

Fig. 5 Network configurations (blue arcs) obtained using DDIP and MIP on a sample of Unit Disk input graphs: the output graphs with maximum possible efficiency, μ, with a given limit on the number of active connections $|E|$. Gray arcs are the allowable arcs which were not used by a solution

The results in Table 3 indicate that the DDIP model is able to form networks that contain efficient structures for information propagation. As expected, for each graph size, the average inverse distance values μ for the DDIP model networks were less than the μ values over fully connected graphs of the same size and type. Further, the resulting edge efficiency values η from the DDIP model networks were larger than the η values for the fully connected networks. The DDIP model performed poorly in comparison with the MIP generated networks, which is to be expected as MIP was run under complete



a (DDIP) $n = 10, |S| = 58, |E| = 28$ **b** (MIP) $n = 10, |S| = 58, |E| = 26$

c (DDIP) $n = 15, |S| = 68, |E| = 32$ **d** (MIP) $n = 15, |S| = 68, |E| = 36$

e (DDIP) $n = 20, |S| = 104, |E| = 49$ **f** (MIP) $n = 20, |S| = 104, |E| = 46$

Fig. 6 Network configurations (blue arcs) obtained using DDIP and MIP on a sample of ERGM input graphs: the output graphs with maximum possible efficiency, μ, with a given limit on the number of active connections $|E|$. Gray arcs are the allowable arcs which were not used by a solution

information (in a centralized way), while the DDIP model was run under partial information (in a decentralized way). Note, however, that the DDIP-generated graphs are more balanced than their MIP counterparts (see Figs. 5 and 6), in that in DDIP solutions the nodes are not overloaded, while MIP typically returns a hub-and-spoke type structure that is very efficient at the expense of some node(s) being overloaded. These results, combined with the low average edge ratio e_r values of the DDIP output networks, show

Table 3 DDIP model results for different numbers of actors

Sample Type	Graph Size	Fully Connected			DDIP					MIP Solution		
		Ties ($\|S\|$)	μ	η	Avg Ties ($\|E\|$)	Avg e_r	Avg μ	Avg η	Run time	Ties ($\|E\|$)	μ	η
Unit Disk	$n = 10$	42	0.711	2.236	22.889	0.545	0.479	1.998	30.544	23	0.602	2.896
	$n = 15$	60	0.519	1.281	35.223	0.587	0.38	1.424	112.589	35	0.454	1.791
	$n = 20$	130	0.594	0.998	51.142	0.393	0.334	1.128	317.731	51	0.486	1.812
ERGM	$n = 10$	58	0.822	2.298	25.875	0.446	0.58	2.423	30.217	26	0.629	2.789
	$n = 15$	68	0.616	1.535	35.6	0.523	0.436	1.676	110.573	36	0.52	2.143
	$n = 20$	104	0.581	1.197	46.25	0.444	0.344	1.287	299.727	46	0.48	1.972

that the model was able to form networks with relatively few ties and good connectivity properties.

Conclusion

The topic of spreading information across networks is widely studied in network science. Most works in the field of information passing, as it relates to social network analysis, focus on the spread of influence across networks and utilize centralized methods in order to find network formation solutions. In certain cases, such as deciding on how to form communication networks for autonomous UAVs, the use of centralized methods is not feasible. This paper thus offers a useful and promising approach towards decentralized network formation modeling that leads to the creation of robust networks that contain structural properties suited for information passing.

The results of the DDIP model application showcase that this new modeling approach can be used to create robust networks. It does have limitations: e.g., in its current form, the DDIP model relies on deterministic rules, which limits its flexibility. The DDIP model was initialized with each actor connecting to one randomly chosen neighbor, to see how the DDIP model solutions vary with different initial connection setups. We observed that some of the final networks were not strongly connected. One could mitigate this issue by allowing the actors to run the DDIP model multiple times.

Further potential extensions and improvements over the presented model could focus on adding parameters and rules to respond to the changes in the network proximity structure. From the analysis perspective, allowing the actors to change their physical positions in space (i.e., accounting for movement) and allowing for external changes to the possible edge set (i.e., accounting for adversarial attacks) would indeed offer interesting extensions.

Abbreviations
DDIP: decentralized deterministic information propagation; ERGM: exponential random graph models; MIP: mixed integer programming.

Acknowledgements
Not applicable.

Authors' contributions
All the authors contributed equally in the modeling and writing of the manuscript. All authors read and approved the final manuscript.

Funding
This work was funded in part by the AFRL Mathematical Modeling and Optimization Institute, by the National Science Foundation Award No. 1635611, and by the U.S. Air Force Summer Faculty Fellowship (Granted to the second author by the Air Force Office of Scientific Research). The work of A. Veremyev was supported in part by the U.S. Air Force Research Laboratory (AFRL) award FA8651-16-2-0009.

Availability of data and materials
The codes for generating the sample networks and the model leading to the conclusions of this article are available at https://github.com/perlaabhinavraj/DDIP.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] Department of Industrial and Systems Engineering, University at Buffalo, Buffalo, NY, USA. [2] Department of Industrial Engineering and Management Systems, University of Central Florida, Orlando, FL, USA. [3] AFRL Munitions Directorate, Air Force Research Laboratory, Eglin, FL, USA.

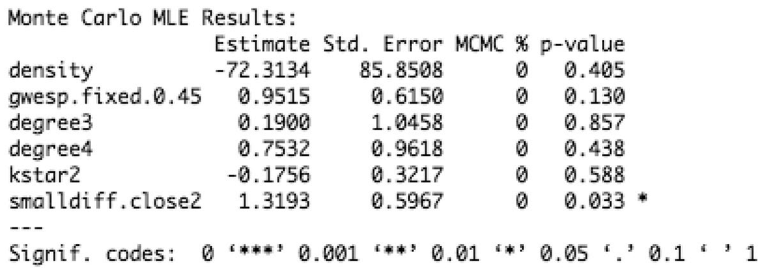
Appendix

ERGM sample graph settings

For each graph size n, an ERGM model was created and fitted to an "observed" graph that was built manually. For each n, the ERGM estimation procedure had to choose fitting parameters based on the structure of the underlying observed input graph. Figures 7 and 8 show the ERGM statistics that were fitted as well as the corresponding parameter estimates for each graph size.

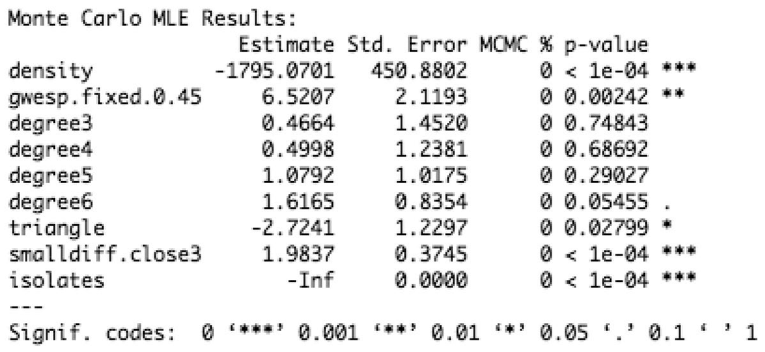
For $n = 10$, there was not much information for the ERGM model to fit parameters to, because of how small the input graph was. There was only one parameter deemed significant, capturing the difference between closeness attributes of connected actors.

Figure 8 describes the set of parameters that were fitted to the input graph of size $n = 20$. The ERGM model for this graph size was able to find some relatively useful information from the input graph. It was found that density, the gwesp statistic, the number of triangles, the difference between "closeness" attributes, and the number of isolates to be significant. Based on these values, the ERGM simulation generated graphs with a low density, high propensity to share connections with connected neighbors, a low number of triangles, no isolates, and more connections between neighbors of similar "closeness" values.



```
Monte Carlo MLE Results:
                   Estimate Std. Error MCMC % p-value
density            -72.3134    85.8508      0   0.405
gwesp.fixed.0.45     0.9515     0.6150      0   0.130
degree3              0.1900     1.0458      0   0.857
degree4              0.7532     0.9618      0   0.438
kstar2              -0.1756     0.3217      0   0.588
smalldiff.close2     1.3193     0.5967      0   0.033 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fig. 7 ERGM parameters for 10 actor graph size



```
Monte Carlo MLE Results:
                     Estimate Std. Error MCMC % p-value
density            -1795.0701   450.8802      0 < 1e-04 ***
gwesp.fixed.0.45      6.5207     2.1193      0 0.00242 **
degree3               0.4664     1.4520      0 0.74843
degree4               0.4998     1.2381      0 0.68692
degree5               1.0792     1.0175      0 0.29027
degree6               1.6165     0.8354      0 0.05455 .
triangle             -2.7241     1.2297      0 0.02799 *
smalldiff.close3      1.9837     0.3745      0 < 1e-04 ***
isolates                -Inf     0.0000      0 < 1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fig. 8 ERGM parameters for 20 actor graph size

Algorithm pseudocodes

Algorithm 2 describes the flow of the DDIP model.

Algorithm 2 The DDIP Model Logic

```
1: $g^t = g^0$;    /*Initialize the output graph at period 0*/
2: for actor $i \in g^0$ do
3:   initialize $P_i$;    /*Initialize the set of actors in the proximity of actor $i$*/
4:   $d_i = \{\}$;    /*Initialize an empty dictionary to store the decisions of actor $i$, where 'key' of
   the dictionary is the decision to take and the 'value' is the happiness for the decision*/
5: end for
6: for actor $i \in g^0$ do
7:   if $g_i^0 = \emptyset$ then
8:     Select $j \in P_i$ w.p. $1/|P_i|$;    /*Have any unconnected actor(s) connect to a possible neigh-
   bor*/
9:     $I_i^0 \leftarrow j$;
10:    $O_j^0 \leftarrow i$;
11:  end if
12: end for
13: for $t \leftarrow 0$ to $t_{max}$ do
14:   for actor $i \in g^t$ do
15:     Propagate $m$ messages;    /*Simulate Message Passing*/
16:     Update $p_{ijk}^t$;    /*Calculate the proportions of messages received in period $t$*/
17:     Calculate $\beta$'s and $h_i(g_i^t)$;    /*Using $p_{ijk}^t$, calculate weights and happiness for actor $i$*/
18:     for Possible Neighbor $j \in P_i \setminus I_i^t$ do
19:       Calculate $h_i(g_{i\leftarrow j}^t)$;    /*Estimate objective after connecting to $j$*/
20:       $d_i \leftarrow ("j" : h_i(g_{i\leftarrow j}^t)) \cup d_i$;    /*Add decision and happiness to decision dictionary*/
21:     end for
22:     for actor $j \in I_i^t$ do
23:       Calculate $h_i(g_{i\not\leftarrow j}^t)$;    /*Estimate happiness after removing active neighbor $j$*/
24:       $d_i \leftarrow ("j" : h_i(g_{i\not\leftarrow j}^t)) \cup d_i$;    /*Add decision and happiness to decision dictionary*/
25:     end for
26:     $d_i \leftarrow ("i" : h_i(g_i^t)) \cup d_i$;    /*Add 'do nothing' decision to decision dictionary*/
27:   end for
28:   for actor $i \in g^t$ do
29:     if $\max(d_i, key = d_i.get) \in I_i^t$ then    /*Find the Max happiness among all decisions*/
30:       $g^t \leftarrow g^t - (j, i)$;    /*If the best move is to subtract edge, remove edge, if allowed*/
31:       $I_i^t \leftarrow I_i^t - j$    /*Remove $j$ from $i$'s in-directed neighbors*/
32:       $O_j^t \leftarrow O_j^t - i$    /*Remove $i$ from $j$'s out-directed neighbors*/
33:     else if $\max(d_i, key = d_i.get) \in P_i \setminus I_i^t$ then
34:       $g^t \leftarrow (i, \max(d_i, key = d_i.get)) \cup g^t$;    /*Best decision is to add a tie to the graph*/
35:       $I_i^t \leftarrow I_i^t + j$    /*Add $j$ to $i$'s in-directed neighbor set*/
36:       $O_j^t \leftarrow O_j^t + i$    /*Add $i$ to $j$'s out-directed neighbor set*/
37:     else
38:       continue    /*Actor $i$ choses to do nothing this period*/
39:     end if
40:   end for
41:   $p_{ijk}^t \leftarrow 0 \ \forall \ i, j, k$;    /* Reset message proportion variables*/
42:   $d_i \leftarrow \emptyset \ \forall i$;    /*Reset decision dictionary for all actors*/
43:   $t \leftarrow t + 1$;    /*Move to next period*/
44: end for
45: Return $g^{tmax}$
```

Algorithm 3 Nelder-Mead Parameter Optimization Algorithm for Decentralized Network Formation Model

```
1: $\mathbf{X} \leftarrow \{x_i^{a,v} | i = 1..n+1,\}$   /*Initialize Parameter Set $\mathbf{X}$ to Be Used*/
2: $t_{max} \leftarrow t_0$   /*Initialize Stalling Termination Parameter $t_{max}$*/
3: $t_{stall} \leftarrow 0$   /*Initialize Stall Count Parameter*/
4: $x^* \leftarrow 0$   /*Initialize best Value $x^*$ to 0*/
5: while $t_{stall} < t_{max}$ do
6:   for $x_i$ in $X$ do
7:     for $p$ in $P$ do
8:       Run Formation Model on $x_i$ using sample graph $p$   /*Use $x_i$'s to obtain $f(x_i)$*/
9:       $x_i^v \leftarrow x_i^v \cup f(x_i)$   /*Add to $x_i$'s saved values*, and Add to the count of values for $x_i$*/
10:      $x_i^a \leftarrow 1/|v| \sum_v (x_i^v)$   /*Update the Average of Parameter $x_i$*/
11:    end for
12:  end for
13:  Sort $X$ by increasing value of $x_i^a$, reassign $i$   /*($x_1$ largest, $x_{n+1}$ smallest)*/
14:  $x_0 \leftarrow \frac{1}{n} \sum_{i=1}^{n} x_i$   /*Calculate the centroid of the parameter space*/
15:  $x_r \leftarrow x_0 + \alpha(x_0 - x_{n+1})$   /*Calculate Reflective Parameter Point $x_r$*/
16:  $x_r^a, \leftarrow f(x_r)$   /*Run model on $x_r$, obtain outcome*/
17:  if $x_r^a < x_1^a$ and $x_r^a >= x_n^a$ then
18:    $x_{n+1} \leftarrow x_r$
19:    if $x_1^a > x^*$ then
20:      $x^* \leftarrow x_1^a$   /*Save new best value*/
21:      $t_{stall} \leftarrow 0$   /*Reset Stalling Count Parameter*/
22:    else
23:      $t_{stall} \leftarrow t_{stall} + 1$   /*Increment the Stalling Count Parameter*/
24:    end if
25:    continue   /*Go back to While Loop*/
26:  else if $x_r^a > x_1^a$ then
27:    $x_e \leftarrow x_0 + \gamma(x_r - x_0)$   /*Calculate Expansion Parameter Point $x_e$*/
28:    $x_e^a, \leftarrow f(x_e)$   /*Run model on $x_e$, obtain outcome*/
29:    if $x_e^a > x_r$ then
30:      $x_{n+1} \leftarrow x_e$
31:      if $x_e^a > x^*$ then
32:        $x^* \leftarrow x_e^a$   /*If $x_e^a$ is the new global Min, update values*/
33:        $t_{stall} \leftarrow 0$
34:      else
35:        $t_{stall} \leftarrow t_{stall} + 1$
36:      end if
37:      continue   /*Go back to While Loop*/
```

The first section of the Nelder–Mead algorithm shown above details how parameters are evaluated and searched for. The second section shows how the contraction and shrink procedures are calculated. The model was implemented using the standard values for the parameters: $\sigma = .5, \rho = .5, \gamma = 2$ and $\alpha = 1$.

Algorithm 4 Nelder-Mead Algorithm Part 2

```
38:         else
39:             $x_{n+1} \leftarrow x_r^a$
40:             if $x_r^a > x^*$ then
41:                 $x^* \leftarrow x_r^a$        /*If $x_r^a$ is the new global Min, update values*/
42:                 $t_{stall} \leftarrow 0$
43:             else
44:                 $t_{stall} \leftarrow t_{stall} + 1$
45:             end if
46:             continue      /*Go back to While Loop*/
47:         end if
48:     else
49:         $x_c \leftarrow x_0 + \rho(x_{n+1} - x_0)$        /*Calculate Contraction Parameter Point $x_c$*/
50:         $x_c^a,\ x_c^v \leftarrow f(x_c)$             /*Run model on $x_c$, obtain value*/
51:         if $x_c^a > x_{n+1}$ then
52:             $x_{n+1} \leftarrow x_c$
53:             if $x_1^a > x^*$ then
54:                 $x^* \leftarrow x_1^a$        /*If $x_1^a$ is the new global Min, update values*/
55:                 $t_{stall} \leftarrow 0$
56:             else
57:                 $t_{stall} \leftarrow t_{stall} + 1$
58:             end if
59:             continue
60:         else
61:             for $i > 1$, and$i < n + 1$ do
62:                 $x_i = x_1 + \sigma(x_i - x_1)$       /*Calculate Shrink Parameter Points*/
63:             end for
64:             if $x_1^a > x^*$ then
65:                 $x^* \leftarrow x_1^a$        /*If $x_1^a$ is the new global Min, update values*/
66:                 $t_{stall} \leftarrow 0$
67:             else
68:                 $t_{stall} \leftarrow t_{stall} + 1$
69:             end if
70:             continue
71:         end if
72:     end if
73: end while
74: return $x_1$
```

Received: 12 April 2019 Accepted: 4 November 2019
Published online: 27 November 2019

References

1. Akyildiz I, Kasimoglu I. Wireless sensor and actor networks: research challenges. Ad Hoc Netw J. 2004;2(4):351.
2. Beard RW, McLain TW, Nelson DB, Kingston D, Johanson D. Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs. Proc IEEE. 2006;94(7):1306.
3. Bala V, Goyal S. Learning from neighbours. Rev Econ Stud. 1998;65(3):595.
4. Gale D, Kariv S. Bayesian learning in social networks. Games Econ Behav. 2003;45(2):329.
5. Villatoro D, Sabater-Mir J, Sen S. Robust convention emergence in social networks through self-reinforcing structures dissolution. ACM Trans Auton Adapt Syst. 2013;8(1):2:1. https://doi.org/10.1145/2451248.2451250.
6. Safar M, Mahdi K, Torabi S. Network robustness and irreversibility of information diffusion in Complex networks. J Comput Sci. 2011;2(3):198. https://doi.org/10.1016/j.jocs.2011.05.005.
7. Jiang C, Chen Y, Liu K. Evolutionary dynamics of information diffusion over social networks. IEEE Trans Signal Process. 2014;62(17):4573. https://doi.org/10.1109/TSP.2014.2339799.
8. Greenan CC. Diffusion of innovations in dynamic networks. J R Stat Soc Series A. 2015;178(1):147. https://doi.org/10.1111/rssa.12054/pdf.
9. Pandit S, Yang Y, Chawla N. Maximizing information spread through influence structures in social networks. In: 2012 IEEE 12th international conference on data mining workshops (ICDMW); 2012. p. 258–65. https://doi.org/10.1109/ICDMW.2012.140
10. Snijders TA, Van de Bunt GG, Steglich CE. Introduction to stochastic actor-based models for network dynamics. Social Netw. 2010;32(1):44.

11. Melodia T, Pompili D, Gungor VC, Akyildiz IF. A Distributed coordination framework for wireless sensor and actor networks. In: Proceedings of the 6th ACM international symposium on mobile Ad Hoc networking and computing. New York: ACM; 2005 MobiHoc '05, p. 99–110. https://doi.org/10.1145/1062689.1062704.
12. Samadi M, Nikolaev A, Nagi R. A subjective evidence model for influence maximization in social networks. Omega. 2016;59:263.
13. Farasat A, Nikolaev AG. Social structure optimization in team formation. Comput Op Res. 2016;74:127.
14. Wu XM, Li Z, So AM, Wright J, Chang SF. Learning with partially absorbing random walks. In: Advances in neural information processing systems; 2012, p. 3077–85. http://papers.nips.cc/paper/4833-learning-with-partially-absorbing-random-walks.
15. De J, Zhang X, Cheng L. Transduction on directed graphs via absorbing random walks. arXiv preprint arXiv:1402.4566; 2014;1402:4566.
16. Kan Z, Yucelen T, Doucette E, Pasiliao E. A finite-time consensus framework over time-varying graph topologies with temporal constraints. J Dyn Syst Meas Control. 2017;139(7):071012.
17. Olfati-Saber R, Fax A, Murray RM. Consensus and cooperation in networked multi-agent systems. Proc IEEE. 2007;95(1):215.
18. Smith B, Egerstedt M, Howard A. Automatic deployment and formation control of decentralized multi-agent networks. In: IEEE international conference on robotics and automation, 2008. ICRA; 2008, p. 134–139. https://doi.org/10.1109/ROBOT.2008.4543198.
19. Lanham M, Morgan G, Carley K. Social network modeling and agent-based simulation in support of crisis de-escalation. IEEE Trans Syst Man Cyber Syst. 2014;44(1):103. https://doi.org/10.1109/TSMCC.2012.2230255.
20. Chinowsky P, Diekmann J, Galotti V. Social network model of construction. J Constr Eng Manag. 2008;134(10):804. https://doi.org/10.1061/(ASCE)0733-9364(2008)134:10(804).
21. Watts A. A dynamic model of network formation. Games Econ Behav. 2001;34(2):331. https://doi.org/10.1006/game.2000.0803.
22. Jackson MO, Watts A. The evolution of social and economic networks. J Econ Theory. 2002;106(2):265. https://doi.org/10.1006/jeth.2001.2903.
23. Snijders TA. The statistical evaluation of social network dynamics. Sociol Methodol. 2001;31(1):361.
24. Sabattini L, Secchi C, Chopra N. Decentralized control for maintenance of strong connectivity for directed graphs. In: 21st mediterranean conference on control and automation; 2013, p. 978–86. https://doi.org/10.1109/MED.2013.6608840
25. Diaz C, Nikolaev A, Pasiliao E. A Decentralized deterministic information propagation model for robust communication. In: International conference on computational social networks. Springer; 2018, p. 235–46.
26. Nikolaev AG, Razib R, Kucheriya A. On efficient use of entropy centrality for social network analysis and community detection. Soc Netw. 2015;40:154.
27. Kemeny JG, Snell JL. Others, Finite markov chains, vol. 356. Princeton: van Nostrand Princeton; 1960.
28. Latora V, Marchiori M. Efficient behavior of small-world networks. Phys Rev Lett. 2001;87(19):198701.
29. Nelder JA, Mead R. A simplex method for function minimization. Comput J. 1965;7(4):308. https://doi.org/10.1093/comjnl/7.4.308.
30. Snijders TAB. Markov chain monte Carlo estimation of exponential random graph models. J Soc Str. 2002;3(2):1.
31. Mukherjee T, Veremyev A, Kumar P, Pasiliao E Jr. The minimum edge compact spanner network design problem. 2017. arXiv preprint arXiv:1712.04010
32. Veremyev A, Prokopyev OA, Pasiliao EL. Critical nodes for distance-based connectivity and related problems in graphs. Networks. 2015;66(3):170.
33. L. Gurobi Optimization. Gurobi optimizer reference manual. 2018. http://www.gurobi.com

Publisher's Note