Learning What Makes A Society Tick

Hung-Ching (Justin) Chen Mark Goldberg Malik Magdon-Ismail
William A. Wallace
Rensselaer Polytechnic Institute (RPI)
Troy, NY 12180, USA
{chenh3, goldberg, magdon}@cs.rpi.edu, wallaw@rpi.edu

Abstract

We present a machine learning methodology (models, algorithms, and experimental data) to discovering the agent dynamics that drive the evolution of the social groups in a community. We use a parameterized probabilistic agent-based model integrating with micro-laws *to present the agent dynamics. The* micro-laws *with different parameters present different actors' behaviors. Our approach is to identify the appropriate parameters in the model including discrete parameters together with continues parameters. To solve this mixed optimization problem, we develop heuristic expectation-maximization style algorithms for determining the appropriate* micro-laws *of a community based on either the observed social group evolution, or observed set of communications between actors without considering the semantics. Also, in order to avoid the resulting combinatorial explosion, we appropriately approximate and optimize the objective within a coordinate-wise gradient ascent (search) setting for continuous (discrete) variables. Finally, we present the learning performance from extensive experiments.*

1 Introduction

"*What makes a society tick?*" is an interesting question. A society of any realm is constituted by agents or *actors* (an actor is commonly referred to an individual entity in the society). Very often, these actors or agents bear a diverse range of behaviors and patterns. Therefore, in a given society, if we are able to understand its actors' behaviors, we should be able to trace the elements (constituents) that make the society ticks. In this study, we uses the group membership and the dynamics of social groups to present the social evolution of a community. In [17], a social group is defined as a collection of actors sharing some common context. The dynamics of social groups is controlled by actors' actions – actors join new groups, leave groups and change groups. An actor's actions are governed by *micro-laws*[1], such as personal attributes (*e.g.*, some people like to go out with a bunch of people, but some would prefer one to one), the actions of other actors (*e.g.*, the actor may join a group because his/her friend is a member of that group), and the influence of the community (*e.g.*, some people take an action from the expectation of the community, but some are not). Sometimes, the same actors, but within different communities, perform different behaviors or are governed by different *micro-laws*. It is an interesting hunt as how the *micro-laws* govern the actors behaviors and make a society ticks.

Recently, the explosion of online communities rises the weight on recognizing how communities evolve. Decision making can be more efficient, accurate, and easier if people can perceive how the community is going to change (*e.g.*, the community is growing larger or smaller), or how the consumers behave. Through this search, we bear a goal to find that "*If given a society's history, can we deduce something about the 'nature' of the society or can we predict the society's future?*" But, what is the "nature" of the society? the "nature" of the society can be – "Do actors generally have an inclination to join small groups or large groups?", "How long does an actor stay in a group?", "How many groups does an actor is in active at same time?", "How do communities influence actors' behaviors?" or "How many social groups are there and what is their average size after 3 months?" and ... etc.

The works on social network analysis have become more popular in the last few years. There is significant literature on modeling of social networks and social network analysis. A lot of works focus on modeling of evolving social networks, *e.g.*, [2, 5, 6, 8, 10, 11, 12, 13, 15, 16, 17]. People try to analyze the social networks from different perspectives using various approaches and different machine learning techniques. In [16], the authors address very simple models fitting in a very limited setting to analyze the

[1] An actor's behaviors have been referred to as *micro-laws* in [9].

evolution of a society. [14] proposes a model for relational data and try to find the hidden structures responsible for the observed autocorrelation among class labels. In [1], the authors use decision-tree approach to determine some properties of the social network. [4] presents a framework to analyze dynamic social networks and try to learn when social interactions occur.

We propose a machine learning methodology (models, algorithms, and experimental data) to discover the agent dynamics that drive the evolution of the social groups in a community. We use the same agent-based social group model introduced in [7], and our presentation of this model follows the one in [7]. The main contribution of the paper is the presentation about the framework of the whole learning process. In the model, there are a lot of different parameters to control different *micro-laws*. We can categorize those parameters into three basic kinds – independent discrete parameters, independent continuous parameters, and dependent discrete parameters. For each kind of parameters, we develop an approach to learn the appropriate values in the *micro-laws*.

Paper Organization. Next, we give an overview of the agent-based probabilistic social group evolution model in Section 2. Then, we present the framework of our approach to learn the appropriate *micro-laws* in the model in Section 3. We discuss what we are going to predict in Section 4. We then give experimental results in Section 5 and discussion in Section 6.

2 Overview of Social Group Model

We give a brief overview of the agent-based probabilistic evolving social group model, called ViSAGE (Virtual Simulation and Analysis of Group Evolution), and [7] presents more details about the model. Figure 1 shows the general framework for the process about the evolution in the social group model at each time step. In the current model, there are actors, groups, the *state* of the society (defined by the properties of actors and groups), and three kinds of actions – *join a group*, *leave a group*, and *do nothing*. According to the *state* of the society, actors decide and perform their actions, and depending on the feedbacks from actors' actions, the properties of actors and groups are updated accordingly.

2.1 Properties of Actors and Groups

Each actor has its own *type*, *rank*, *qualification*, and *resource*. An actor's *type* control the actor's group size preferences (what size of group the actor most likely join or leave) and the actor's "ambition" (how quickly the actor's rank increases in a group). An actor's *rank* presents the actor's position in the group. The rank is directly proportional to

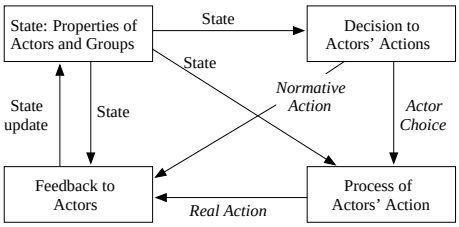

Figure 1. Framework for the process about the evolution in the social group model at each time step.

the amount of time that an actor has been a member of that group, and the rank also depends on the actor's type. An actor's *qualification* presents an actor's prestige. The qualification is determined as the average rank of the actor among all the groups the actor has been a member. Each actor has its available *resource* which depends on how many resource an actor needs to maintain a membership in a group, and the actor's resource influence what kind of action an actor can complete at next time step.

The properties of groups include *qualification* and *membership*. Each group has its qualification, defined as the average qualification of actors currently participating in the group, and it influences the acceptance of actors' application to join. Group's *membership* indicates which actors are joining the group.

2.2 Actors' Actions

Every actor needs to decide to leave one group, join one group, or remain in the same groups at each time step. The decision depends on an actor's available resource. After an actor has chosen which action it would like to perform, it needs to decide which group to join or leave. Each actor has a size preference and qualification; therefore, the actor takes into account the size and qualification of the group during decision making. The probability of which group the actor decides to join or leave depends on the group's membership and qualification and also the actor's ranks, size preference and qualification.

2.3 State Update

At the end of each time step, the last step of the process is to update the properties of actors and groups. To update properties of actors and groups is based on all actors' actions and the society reward/penalty parameters θ_{reward}. The θ_{reward} determine how to update an actor's resource, and it is summarized heuristically by

$Reward\left(action, R_E^1, \theta_{action}, \theta_{reward}\right)$, where θ_{action} indicates some parameters related to actors' action.

2.4 Example

Figure 2 illustrates an example of an evolving community from time t to $t+1$, and it is easily extends to an arbitrary number of actors and groups. In this example, there

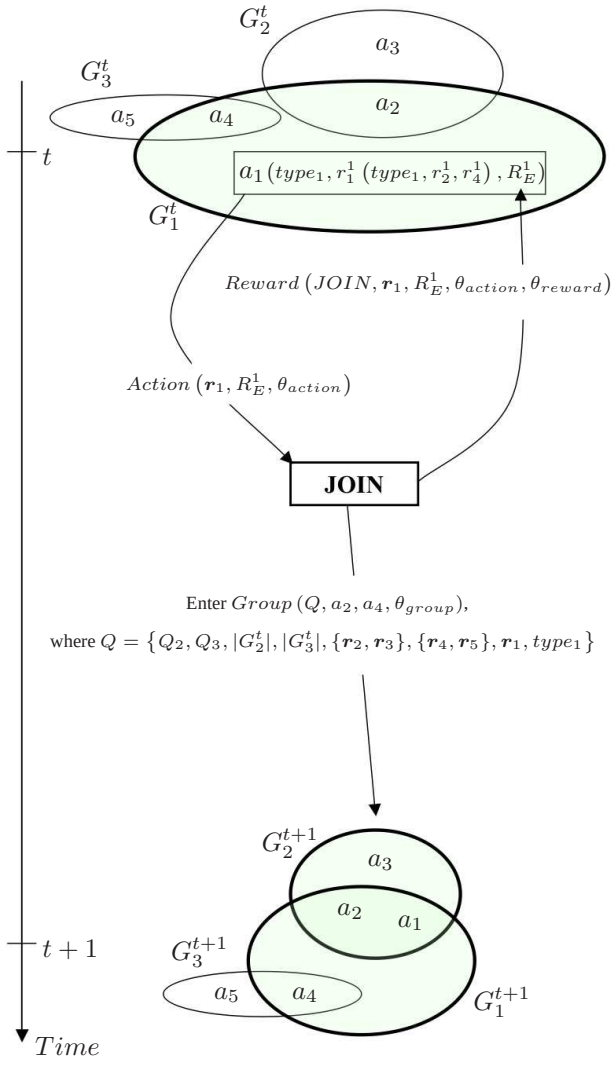

Figure 2. An example of the agent-based probabilistic social group evolution model

are five actors, a_1, a_2, a_3, a_4 and a_5, and three social groups G_1, G_2 and G_3 at time t and $t+1$. Focus on actor a_1 at time t. Some a_1's properties have been indicated: type ($type_1$), ranks ($\boldsymbol{r}_1$), resource (R_E^1). As indicated, r_1^1 depends on $type_1$ and other actors' ranks (r_2^1 and r_4^1) in the group; thus r_1^1 indirectly depends on $type_2$ and $type_4$. According to the actor's properties, a_1 decides to join a new group through the stochastic process denoted by $Action()$ in Fig. 2, which depends on a set of parameters θ_{action}; two other possible actions are to leave a group or to do nothing.

In this example, a_1 decides to join a group and must now decide which specific group to join. This is accomplished by a stochastic hand-shaking process in which a_1 decides which group to "apply" to, G_2 in this case, and G_2 decides whether or not to accept a_1 into the group. This process is indicated by $Group()$ in Fig. 2 and is governed by its own set of parameters θ_{group}, together with the properties of some of the other actors and the group structure. Actor a_1 learns from its *neighbors* a_2, a_4 about which other groups are out there to join; the potential joining groups are G_2 and G_3 in this case. In the example, a_1 applied to G_2 and in this particular case was accepted.

The resource of a_1 now get updated through a stochastic process denoted by $Reward()$, which additionally depends on the actual action and some parameters θ_{reward}. This process is analogous to society encouragement or penalty for doing expected versus unexpected things.

After all actors go through a similar process in batch mode and independently of each other, the entire state of the society is updated in a feedback loop as indicated in Fig. 1 to obtain the state at time $t+1$.

3 Learning Process

Our approach is to identify the appropriate parameters in the model from either the social group evolution, or the communication data between actors without considering the content of the messages. If we have the communication data, we need to discover the group evolution from the communications, and then learn from the group evolution. From the group evolution, we are able to determine the actor dynamics and learn the appropriate parameters in the model.

3.1 Learning From Communications

The challenge with real data is that the groups and their evolution are not known, especially in online communities; instead, one observes the communication dynamics. We assume that a pair of actors in many groups together are more likely to communicate with each other often; therefore, the communication dynamics are able to indicate the group dynamics. The first step in learning is to use the communication dynamics to construct the set of groups and convert the problem to one of learning from groups as in Section 3.2.

Imagine that communications between the actors are aggregated over some time period τ to obtain a weighted communication graph G_τ. The actors are nodes in G_τ and the edge weight w_{ij} between two actors is the communication intensity (number of communications) between i and j. The

sub-task we would like to solve is to infer the group structure from the communication graph G_τ. Any reasonable formulation of this problem is NP-hard, and so we need some efficient heuristic for finding the clusters in a graph that correspond to the social groups. In particular, the clusters should be allowed to overlap, as is natural for social groups. This excludes most of the traditional clustering algorithms, which partition the graph. We use the algorithms developed by Baumes et al. [3], which efficiently find overlapping communities in a communication graph.

We consider time periods $\tau_1, \tau_2, \ldots, \tau_{T+1}$ and the corresponding communication graphs $G_{\tau_1}, G_{\tau_2}, \ldots, G_{\tau_{T+1}}$. Given a single graph G_{τ_t}, the algorithms in [3] output a set of overlapping clusters, which we then use as the data $\mathcal{D}_t$ (the social group structure at time step t). However, in order to use the learning prescription as in Section 3.2, one needs to construct the paths of each actor. This means we need the correspondence between groups of time step t and $t+1$. Formally, we need a matching between the groups in $\mathcal{D}_t$ and $\mathcal{D}_{t+1}$ for $t = 1, 2, \ldots, T-1$: for each group in $\mathcal{D}_t$, we must identify the corresponding group in $\mathcal{D}_{t+1}$ to which it evolved. If there are more groups in $\mathcal{D}_{t+1}$, then some new groups arose. If there are fewer groups in $\mathcal{D}_{t+1}$, then some of the existing groups disappeared. In order to find this matching, we use a standard greedy algorithm.

Finding Matchings Let $\mathcal{X} = \{X_1, \ldots, X_n\}$ and $\mathcal{Y} = \{Y_1, \ldots, Y_n\}$ be two collections of sets, and we allow some of the sets in $\mathcal{X}$ or $\mathcal{Y}$ to be empty. We use the symmetric set difference $d(x, y) = 1 - |x \cap y|/|x \cup y|$ as a measure of error between two sets. Then, we consider the complete bipartite graph on $(\mathcal{X}, \mathcal{Y})$ and would like to construct a matching of minimum total weight, where the weight on the edge (X_i, Y_j) is $d(X_i, Y_j)$.

3.2 Learning From Group Evolution

We first introduce some notation. The set of actors is $\mathcal{A}$; we use i, j, k to refer to actors. The data $\mathcal{D} = \{\mathcal{D}_t\}_{t=1}^{T+1}$ is the set of social groups at each time step, where each $\mathcal{D}_t$ is a *set* of groups, $\mathcal{D}_t = \{G_l^t\}_l$, $G_l^t \subseteq \mathcal{A}$; we use l, m, n to refer to groups. Collect all the parameters which specify the model as $\mathbf{\Theta}_M$, which includes all the parameters specific to an actor (*e.g.* $type$) and all the global parameters in the model (*e.g.* $\theta_{action}, \theta_{reward}, \theta_{group}$). We would like to maximize the likelihood

$$\mathcal{L}(\mathbf{\Theta}_M) = Prob(\mathcal{D}|\mathbf{\Theta}_M). \quad (1)$$

We are able to determine all actors' actions based on the group evolution; then, We define the path of actor i, $\mathbf{p}_i^T = (p_i(1), \ldots, p_i(T))$, as the set of actions it took over the time steps $t = 1, \ldots, T$. The actions at time t, $p_i(t)$, constitute deciding to join, leave or stay in groups, as well as which groups were left or joined. Given $\mathcal{D}$, we can construct $\mathbf{p}_i^T$ for every actor i, and conversely, given $\{\mathbf{p}_i^T\}_{i=1}^{|\mathcal{A}|}$, we can reconstruct $\mathcal{D}$. Therefore, we can alternatively maximize

$$\mathcal{L}(\mathbf{\Theta}_M) = Prob(\mathbf{p}_1^T, \ldots, \mathbf{p}_{|\mathcal{A}|}^T \| \mathbf{\Theta}_M). \quad (2)$$

It is this form of the likelihood that we manipulate. Typical ways to break up this optimization is to iteratively first improve the continuous parameters and then the combinatorial (discrete) parameters.

The main problem we face is an algorithmic one, namely that typically, the number of actors, $|\mathcal{A}|$, is very large (thousands or tens of thousands), as is the number of time steps, T, (hundreds). Another problem is that in our model, some actor's properties are independent to others' properties, but also some actor's properties are dependent to others' properties, causing dimensionality curse.

For independent parameters, the maximization over a single actor's parameters only involves that actor's path and is a factor of $|\mathcal{A}|$ more efficient to compute than if we looked at all the actor paths. Therefore, the entire learning process can be summarized by maximizing over each parameter successively, where to maximize over the parameters specific to an actor, we use only that actor's path. We develop an greedy search algorithm[2] to learn the discrete parameters, and use a gradient based approach to optimize the independent continuous parameters, which involves taking derivatives of equation (2).

For dependent parameters, there is only dependent discrete parameters in our model. The values of an actor's parameters depend on all other actors' parameters; in other words, the maximization involves all actors' paths. If there are N possible values for a certain parameter, the complexity to find the globe maximum is $\Omega(N^{T|\mathcal{A}|})$. We know $|\mathcal{A}|$ and T are very large, and the computation is really time consuming. we develop an approximate expectation-maximization (EM) style algorithm to learn the appropriate values. The first step, we learn what are the probabilities of all possible values for each actor. Then, based on those probabilities, we randomly generate all actors' paths and iteratively find the approximate values.

4 Prediction

Unlike in a traditional supervised learning task, where the quality of the learner can be measured by its performance on a test set, in our setting, the learned function is a stochastic process, and the test data are a realization of the stochastic process. Specifically, assume we have training data $\mathcal{D}_{train} = \{\mathcal{D}_t\}_{t=1}^{T+1}$ and test data $\mathcal{D}_{test} = \{\mathcal{D}_t\}_{t=T+2}^{T+K}$.

[2]The detail of the greedy search algorithm learning independent discrete parameters can be found in [7]

We learn the parameters governing the micro-laws using $\mathcal{D}_{train}$, and use multi-step prediction to test on the test data. Specifically, starting from the social group structure $\mathcal{D}_{T+1}$ at time $T+1$, we predict the actions of the actors, i.e. the actor paths into the future. Based on these paths, we can construct the evolving social group structure and compare these predicted groups with the observed groups on the test data using some metric. Here, we use the distribution of group sizes to measure our performance. Specifically, let $N_k^{pred}(t)$ and $N_k^{true}(t)$ be the number of groups of size k at time t for the predicted and true societies respectively. We report our results using the squared error measure between the frequencies as well as the squared error difference between the probabilities,

$$E_f(t) = \sqrt{\frac{\sum_k \left(N_k^{pred}(t) - N_k^{true}(t)\right)^2}{\sum_k \left(N_k^{true}(t)\right)^2}}. \qquad (3)$$

5 Experiments

In our model, the parameters can be learned using the approach described in Section 3. Here we focus on three parameters, the actor's type, the initial resource allocation, and the society reward/penalty parameters θ_{reward}.

5.1 Learning Actors' Types

Here, we present the results from learning actors' types, an independent discrete parameter. In our model, there are three kinds of actor type – Leader prefers small group size and is most ambitious, Socialite prefers medium group and is medium ambitious, and Follower prefers large group and is less ambitious. To evaluate performance, we use the model to generate simulation data for training and testing. Since we know the values of the parameters in the model, we can compare the true values with the learned values to compute the accuracy. We evaluate the results from the following 6 different algorithms:

- Cluster: For each actor i, let $size_i$ be the average size of groups actor i joined. We cluster $\{size_i\}_{i=1}^{|\mathcal{A}|}$ into 3 groups using the standard 3-means algorithm. Leaders are the smallest, followers the largest and socialites the middle. This is a simple heuristic based on the observation that leaders join small groups and followers large groups.
- Learn: The learning algorithm described in Sect. 3.2.
- Optimal: The ground truth model. (For comparison and only available on simulated data.)
- Leader, Socialite, Follower: benchmarks which assign all actors to leader, socialite or follower types respectively.

Table 1 shows the accuracy of Cluster and Learn using 100 time steps of training data (averaged over 20 data sets). Each data set was created by randomly assigning about 1/3 of the actors to each class and simulating for 100 time steps. All others parameters except types were held fixed.

Accuracy 52.83%		Learned Actors' Types		
		L	*S*	*F*
True Actors' Class	*L*	38.3	93.85	5.25
	S	0.65	75.45	132.4
	F	0.0	3.7	150.4

(a) Cluster algorithm

Accuracy 83.75%		Learned Actors' Types		
		L	*S*	*F*
True Actors' Class	*L*	132.85	4.55	0.0
	S	28.3	166.25	13.95
	F	6.15	28.3	119.65

(b) Learn algorithm

Table 1. Confusion matrix in learning types for Cluster and Learn. For comparision, the accuracy of randomly assigning type is 33%, where accuracy is the % of correctly classified actors. (L = Leader, S = Socialite, F = Follower).

It is clear that Learn algorithm has better performance than Cluster algorithm because Cluster only considers actors' group size preference, but Learn also considers actors' ambition and influences from their friends and the community.

5.2 Impact of Learning on Prediction

Based on the predictive performance, we use the model to determine which parameters are worth learning and which are not. Figure 3 shows the impact of the optimal predictor (all parameters set to their true values) vs. choosing a random society reward/penalty parameters θ_{reward}, and for various choices of actors' initial resource allocation$\{E_i\}$ (every actor assigned some fixed initial resource according to max$\{E_i\}$, min$\{E_i\}$ or average$\{E_i\}$). As can be seen, using the wrong $\{E_i\}$ does not significantly affect performance but the wrong θ_{reward} does - which might be expected as the effects of initial conditions should equilibrate out.

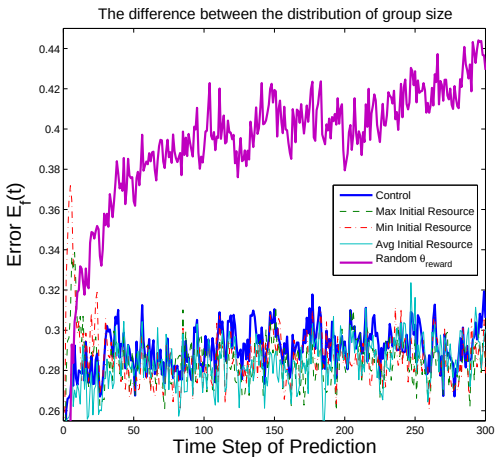

Figure 3. Impact of using incorrect θ_{reward} and $\{E_i\}$ on predictive error.

6 Discussion

There are a lot of stories can be discovered from a community. Our work addresses the problem using a very general setting. While we present our methodology in the context of a specific model, it can be appropriately applied and extended to any parameterized model. The model can be integrated with different *micro-laws* in a society that one wishes to find. Furthermore, by applying the same techniques, one will be able to find the appropriate parameters within the *micro-laws* that make a society ticks.

The results also show some interesting findings. People sometimes change their behavior because the influence from the outside environment, and what they act is different from what they really want to behave. If only based on the observed behavior, such as Cluser algorithm, people can only capture the explicit behaviors. However, through Learn, people not only can capture actors' observable behaviors, but also the influence from the society (*e.g.*, friendship) which enable them to discover actors' implicit behaviors.

Acknowledgment

This material is based upon work partially supported by the National Science Foundation under Grants Nos. IIS-0324947 and CNS-0323324, and No. 0634875, and by the ONR Grant No. N00014-06-1-0466. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or U.S. Government.

References

[1] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: Membership, growth, and evolution. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54, 2006.

[2] J. A. Baum and J. Singh, editors. *Evolutionary Dynamics of Organizations*. Oxford Press, New York, 1994.

[3] J. Baumes, M. Goldberg, and M. Magdon-Ismail. Efficient identification of overlapping communities. In *IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 27–36, 2005.

[4] T. Y. Berger-Wolf and J. Saia. A framework for analysis of dynamic social networks. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 523–528, New York, NY, USA, 2006. ACM Press.

[5] H. Bonner. *Group Dynamics*. Ronald Press Company, New York, 1959.

[6] K. Carley and M. Prietula, editors. *Computational Organization Theory*. Lawrence Erlbaum associates, Hillsdale, NJ, 2001.

[7] H.-C. Chen, M. Goldberg, M. Magdon-Ismail, and W. A. Wallace. Inferring agent dynamics from social communication network. In *Joint 9th WebKDD and 1st SNA-KDD Workshop at KDD*, 2007.

[8] K. Chopra and W. A. Wallace. Modeling relationships among multiple graphical structures. *Computational and Mathematical Organization Theory*, 6(4), 2000.

[9] M. Goldberg, P. Horn, M. Magdon-Ismail, J. Riposo, D. Siebecker, W. Wallace, and B. Yener. Statistical modeling of social groups on communication networks. In *First conference of the North American Association for Computational Social and Organizational Science*, 2003.

[10] P. Holland and S. Leinhardt. Dynamic model for social networks. *Journal of Mathematical Sociology*, 5(1):5–20, 1977.

[11] R. Leenders. Models for network dynamics: A Markovian framework. *Journal of Mathematical Sociology*, 20:1–21, 1995.

[12] T. F. Mayer. Paries and networks: Stochastic models for relationship networks. *Journal of Mathematical Sociology*, 10:51–103, 1984.

[13] P. Monge and N. Contractor. *Theories of Communication Networks*. Oxford University Press, 2002.

[14] J. Neville and D. Jensen. Leveraging relational autocorrelation with latent group models. In *Proceedings of the 5th IEEE International Conference on Data Mining*, pages 322–329, 2005.

[15] M. E. J. Newman. The structure and function of complex networks. *SIAM Reviews*, 45(2):167–256, June 2003.

[16] A. Sanil, D. Banks, and K. Carley. Models for evolving fixed node networks: Model fitting and model testing. *Journal oF Mathematical Sociology*, 21(1-2):173–196, 1996.

[17] S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge University Press, 1994.