Inferring Agent Dynamics from Social Communication Network

Hung-Ching (Justin) Chen
Department of Computer
Science, RPI
Troy, New York 12180
chenh3@cs.rpi.edu

Malik Magdon-Ismail
Department of Computer
Science, RPI
Troy, New York 12180
magdon@cs.rpi.edu

Mark Goldberg
Department of Computer
Science, RPI
Troy, New York 12180
goldberg@cs.rpi.edu

William A. Wallace
Department of Decision
Systems and Engineering
Science, RPI
Troy, New York 12180
wallaw@rpi.edu

ABSTRACT
We present a machine learning approach to discovering the agent dynamics or *micro-laws* that drives the evolution of the social groups in a community. We set up the problem by introducing a parameterized probabilistic model for the agent dynamics: the acts of an agent are determined by *micro-laws* with unknown parameters. Our approach is to identify the appropriate *micro-laws* which corresponds to identifying the appropriate parameters in the model. To solve the problem we develop heuristic expectation-maximization style algorithms for determining the *micro-laws* of a community based on either the observed social group evolution, or observed set of communications between actors. We present the results of extensive experiments on simulated data as well as some results on real communities, *e.g.*, newsgroups.

Keywords
Social Network, Agent-Based Modeling, Machine Learning, EM Algorithm

1. INTRODUCTION
The social evolution of a community can be captured by the dynamics of its social groups. A social group is a collection of agents, or *actors*[1] who share some common context [19]. The dynamics of the social groups are governed by the actor dynamics – actors change groups, leave groups and join new groups. An actor's actions are governed by *micro-laws* which may be: personal attributes, *e.g.* some people are "social" or "outgoing" and others are not; the actions of other actors, *e.g.* the actor may join a group because his/her

---

[1]An actor generically refers to an individual entity in the society.

friend is a member of that group; the social structure in the community, *e.g.* the actor may join a group because it is powerful or "central". In summary, any reasonable model for an agent based evolving community must necessarily involve complex interactions between actors' attributes and the social group structure itself.

What makes one community different from another? Consider the community of college students versus the community of bloggers. The same sorts of people form a large fraction of both of these communities (young adults between the ages of 16-24), yet the group structure in both of these communities is quite different. For example, an actor typically belongs to one college, but may belong to many blog-communities. It is intuitively clear why such significant differences exist between these two communities. For example the process of gaining admission into a social group is considerably harder and more selective for the college community than a typical online community. The *micro-laws* (*e.g.* actors' group size preferences) which govern actors' dynamics are different in these two cases, and hence the resulting communities look quite distinct.

The explosion of online communities provides an ideal pasture on which to groom and test social science theories, in particular the most natural question is: *what are the micro-laws which govern a particular society*? An efficient approach to answering this question for a given community yields a powerful tool for a sociologist, that can be used for discovering the dominant factors that determine how a community evolves.

There are a number of challenges, the first and foremost being the complex nature of the micro-laws needed to represent even a very simple society – actors may have discrete attributes together with continuous parameters, which inevitably leads to a mixed optimization problem, and each actor has his/her own attributes, which also interact with others' attributes, suffering from combinatorial and dimensionality curses. Another challenge is that the data upon which to answer the question is not available – typically social groups (especially online groups) do not announce their membership and one has to infer groups from observable macro-quantities such as communication statistics. Given the recent explosion in online communities, it is now recognized that understanding how

communities evolve is a task of rising importance. We take a machine learning approach to these challenges.

Our Contributions. We present a machine learning methodology (models, algorithms, and experimental data) for determining the micro-laws of a community based on either the observed social group evolution, or observed set of communications between actors; most of time, we are only able to collect communications between actors in a real world. Our approach uses a parameterized probabilistic micro-law based model[2] of social group evolution, and we develop some algorithms to identify the appropriate micro-laws which corresponds to identifying the appropriate parameters in the model. We identify the appropriate micro-laws by solving a mixed optimization problem because our model contains discrete parameters as well as continuous parameters. And to avoid the resulting combinatorial explosion, we appropriately approximate and optimize the objective within a coordinate-wise gradient ascent (search) setting. To test the quality of our approximations and the feasibility of the approach, we present the results of extensive experiments on simulated data as well as some results on real data, *e.g.*, newsgroups.

Related Work. There is significant literature on modeling of social networks and social network analysis [2, 5, 7, 8, 10, 11, 13, 14, 15, 17, 18, 19]. Most of this work focuses on modeling of evolving social networks. In [17] the authors address model fitting in a very limited setting using very simple models. Our work addresses the problem using a much more general setting. While we present our methodology in the context of a specific model, it can be appropriately extended to any parameterized model. In [1], the authors use decision-tree approach to determine some properties of the social network. The decision-tree approach is a deterministic process. This is different from our approach in using the stochastic process to determine actors' behaviors. The reason is that in the social network, even under the same environment, actors do not necessary possess or reflect the same behaviors.

Paper Organization. Next, we give an overview of the probabilistic social group evolution model called ViSAGE [3] in Section 2.Then, we present our approach to learning the appropriate parameters of the model from given the data of groups evolution in Section 3, and followed by learning from communications data, which most of time, we can only collect from real communities, in Section 4. We discuss what we are going to predict in Section 5. Then, we give experimental results on simulation data and real data in Section 6 and conclude in Section 7.

2. OVERVIEW OF SOCIAL GROUP MODEL

We give a brief overview of the probabilistic evolving social group model ViSAGE , details of which can be found in the appendix material . Figure 1 shows the general framework for the step by step evolution in the social group model. In the current model, there are actors, groups, the *state* of the society which is defined by properties of the actors and groups, and three kinds of actions – *join a group*, *leave a group*, and *do nothing*. Based on the *state* of the society, each actor decides which action the actor most likely wants to execute, which is known as the *Normative Action*. Nonetheless, under the influence of the present communities, actions of actors are

affected. After being influenced, each actor eventually performs the *Real Action*. Depending on the feedbacks from actors' *Normative Action* and *Real Action*, properties of actors and groups are updated accordingly.
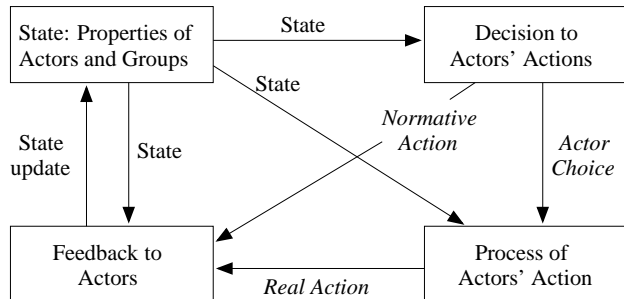
Figure 1: Framework for step by step evolution in the social group model.

2.1 Properties of Actors and Groups

Many parameters govern how actors decide whether to join or leave a group, or do nothing, and also which group the actor desires to join or leave; *i.e.*, parameters such as the group memberships, how long each actor has been in a group, the ranks of the actors and the amount of resources available to an actor (in excess of what is currently being used to maintain the actor's current group memberships). In the following sections, we briefly present some important parameters:

2.1.1 Type

Each actor has a private set of attributes, which we refer to as its *type*. In our current model, the type of an actor simply controls the actor's group size preferences and her "ambition" (how quickly her rank increases in a group). There are 3 kinds of type in the current model:

1. *Leader* who prefers small groups and is the most ambitious.

2. *Socialite* who prefers medium sized groups.

3. *Follower* who prefers large groups and is the least ambitious.

We use notation a_i for actor i and $type_i$ for actor i's type.

2.1.2 Rank

Each actor has a rank in each group to present the actor's position in the group. As actors spend more time in a group, their position in the group changes. There is a tendency for more senior members of a group to have a higher position within the group than junior members. Also, $Leader$ have a higher tendency to increase their position within the group than $Follower$. The definition of rank is as follow:

$$r_i^l = \frac{time_i^l \delta^i}{\sum_{j \in G_l} time_j^l \delta^j}, \qquad (1)$$

where we use notation G_l for social group l, r_i^l for the rank of actor a_i in group G_l, $time_i^l$ for the amount of time that actor i has spent in group G_l, and δ^i indicates how quickly actor i's rank increases in a group each time step, which depends on actor i's type ($type_i$).

2.1.3 Qualification

Each actor has a qualification (q_i) to represent an actor's prestige. It is determined as the average rank of the actor among all the groups the actor has been a member, and the rank is weighted to give a stronger weight to ranks from larger groups. The qualification of an actor is used for an actor to determine which group she more likely join or leave. The definition of qualification is as follow:

$$q_i = \frac{\sum_{i \in G_l} r_i^l |G_l|}{\sum_{i \in G_l} |G_l|}, \tag{2}$$

where we use $|G_l|$ to denote the size of group l, and it means how many actors are in that group.

Similarly, each group has its qualification, defined as the average qualification of actors currently participating in the group. The higher a group's qualification, the more attractive it will appear to other actors looking for a group to join. The qualification of group G_l is defined by the formula

$$Q_l = \sum_{i \in G_l} q_i r_i^l. \tag{3}$$

2.1.4 Actor Resources

We use R_E^i as the available resources for actor a_i and E_i as the initial value of the actor's resources. R_E^i depends on how many resources an actor needs to maintain a membership in a group. In addition, the actors' ranks and the number of groups the actor is in influences how many resources an actor needs to maintain a membership in a group. And R_E^i also influences what kind of action an actor can complete at next time step, described in §2.2.

2.2 Actors' Normative Actions

At each time step, every actor needs to decide on leaving one group, joining one group, or remaining in the same groups. The decision depends on an actor's available resources (R_E^i), where i indicates actor i. If R_E^i is positive, the actor will tend to use the available resources in joining another group. If R_E^i is negative, the actor will tend to leave a group in order to lessen the cost needed. If R_E^i is near zero, the actor may decide to remain in all the same groups for the current time step. Based on the R_E^i, we can determine the probabilities for actor i to join, leave, or do nothing, and we use the notations P_+, P_0, and P_- to denote the probabilities, respectively. And $P_+ + P_0 + P_- = 1$. We define the $Normative$ action as

$$Act_{normative} = \underset{action}{\mathrm{argmax}}(P_x), \tag{4}$$

where $P_x = \{P_+, P_0, P_-\}$.

2.3 Actors' Real Actions

Ideally, the actor would always choose to perform the $Normative$ action, since this creates a state of stability. However, we assume that the actors sometimes make non-rational decisions, regardless of the amount of available resources they have. An actor chooses an action she will perform based on a stochastic process as follow:

$$Act_{choice} = random(P_+, P_0, P_-), \tag{5}$$

where $random(P_+, P_0, P_-)$ means randomly choose an action based on the values of P_+, P_0, and P_-.

After an actor has chosen which action she would like to perform, she needs to decide which group to join or leave. Each type of actors has a size preference; therefore, the actor takes into account the size and qualification of the group during decision making. The size preference is defined by a function of actor's type and groups' size,

$$SizeAff_i^l = Function(type_i, |G_l|), \tag{6}$$

which indicates the size preference for actor i to group l. Then the probability of which group the actor decides to join or leave depends on the qualification of the group and the actor's ranks, size preference and qualification. The probability actor i would like to select group l to join or leave is defined as follow:

$$Probability_i^l = Function(\boldsymbol{r}_i, SizeAff_i^l, q_i, Q_l), \tag{7}$$

where $\boldsymbol{r}_i$ denotes the set of ranks of actor a_i in all her groups. Then, if actor i chooses group l to join, group l can accept or reject actor i's application based a stochastic process, which is related to the group's qualification and the actor's qualification. The probability of group l rejecting actor i is as follow:

$$Prob\_Reject_i^l = Function(q_i, Q_l). \tag{8}$$

2.4 State Update

The final step of the process at each time step is to update the properties of actors and groups. To update properties of actors and groups is based on all actors' $Normative$ actions and real actions and the society reward/penalty parameters θ_{reward}. The reward/penalty parameters θ_{reward} determine how to update an actor's resources, 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' actions.

2.5 Example

Figure 2 gives an illustrative example of an evolving community from time t to $t + 1$. We use this example to describe the essential details of the model, which easily extends to an arbitrary number of actors and groups.

In this example (Fig. 2), there 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 $t + 1$; we use G_l^t for social group l at time t. Focus on actor a_1 at time t. Some of the properties of actor a_1 have been indicated: $type_1, \boldsymbol{r}_1, R_E^1$. As indicated, r_1^1 depends on $type_1$ through how long the actor a_1 has been in the group G_1; it also depends on the ranks of the other actors, a_2 and a_4, in the group, through the fact that the sum of all ranks of actors in a group is 1.

$$\sum_{i \in G_l} r_i^l = 1 \tag{9}$$

Thus r_1^1 indirectly depends on $type_2$ and $type_4$. Based on her 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.

Having decided to join a group, a_1 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 (a_2, a_4 in this case) and the group structure. Actor a_1 learns about which other groups are out there to join
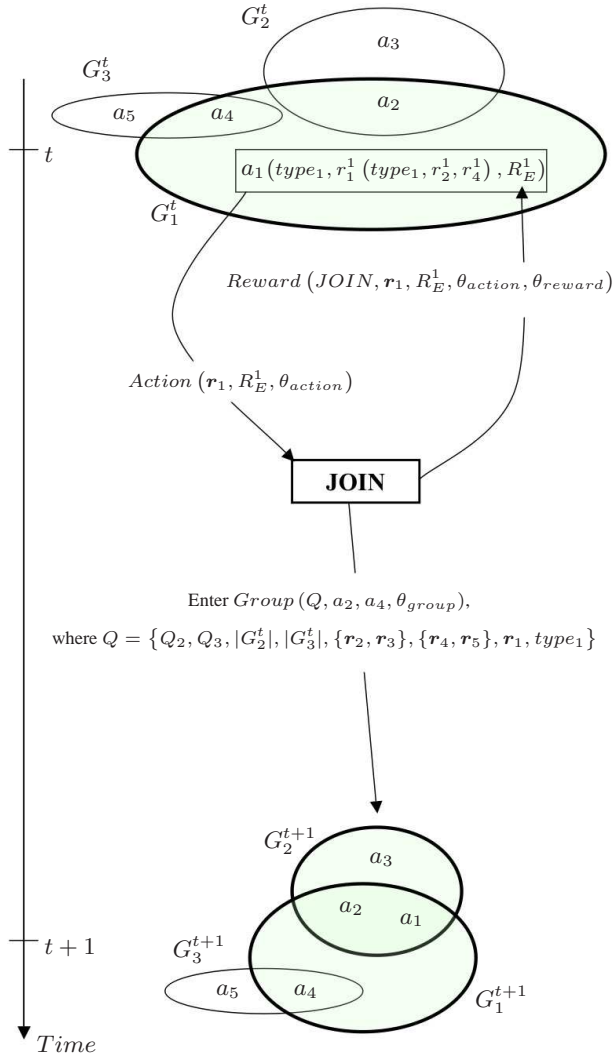
Figure 2: An example of the probabilistic social group evolution model

through its *neighbors* a_2, a_4 – i.e., the groups they belong to, and thus only applies to other groups by *reference*; the potential joining groups are G_2 and G_3 in this case. Actor a_1 then decides which group to apply to based on her *qualification*, as measured by her average rank in all her groups, and the qualification thresholds and sizes of the groups. A group also decides whether to accept a_1 based on similar properties. In the example, a_1 applied to G_2 and in this particular case was accepted.

The resources of a_1 now get updated by some reward, through a stochastic process denoted by $Reward()$ in Fig. 2, 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 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 – Fig. 3 shows an example of training data which includes the information about group evolution. 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

$$\mathcal{L}(\mathbf{\Theta}_M) = Prob(\mathcal{D}|\mathbf{\Theta}_M). \tag{10}$$

We define the path of actor i, $\mathbf{p}_i^T = (p_i(1), \dots, p_i(T))$, as the set of actions it took over the time steps $t = 1, \dots, T$. 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

$$\mathcal{L}(\mathbf{\Theta}_M) = Prob(\mathbf{p}_1^T, \dots, \mathbf{p}_{|\mathcal{A}|}^T \| \mathbf{\Theta}_M). \tag{11}$$

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 continuous parameters can be optimized using a gradient based approach, which involves taking derivatives of $\mathcal{L}(\mathbf{\Theta}_M)$. This is generally straightforward, though tedious, and we do not dwell on the technical details. 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). From the viewpoint of actor i, we break down $\mathbf{\Theta}_M$ into three types of parameters: θ_i, the parameters specific to actor i, in particular its type and initial capital; $\theta_{\bar{i}}$, the parameters specific to other actors; and, θ_G, the parameters of the society, global to all the actors. The optimization style is iterative in the following sense. Fixing parameters specific to actors, one can optimize with respect to θ_G. Since this is a fixed number of parameters, this process is algorithmically feasible. We now consider optimization with respect to θ_i, fixing $\theta_{\bar{i}}, \theta_G$. This is the task which is algorithmically non-trivial, since there are $\Omega(|\mathcal{A}|)$ such parameters.

In our model, the actors at each time step take independent actions. At time t, the state of the society $\mathcal{I}_t$ can be summarized by the group structure, the actor ranks in each group and the actor surplus resources. Given $\mathcal{I}_t$, each actor acts independently at time t. Thus, we can write

$$\mathcal{L}(\mathbf{\Theta}_M) = Prob(\mathbf{p}_1^{T-1}, \dots, \mathbf{p}_{|\mathcal{A}|}^{T-1}|\mathbf{\Theta}_M) \times$$
$$\prod_{i=1}^{|\mathcal{A}|} Prob(p_i(T)|\mathbf{\Theta}_M, \mathcal{I}_T). \tag{12}$$

Continuing in this way, by induction,

$$\mathcal{L}(\mathbf{\Theta}_M) = \prod_i \prod_t Prob(p_i(t)|\mathbf{\Theta}_M, \mathcal{I}_t). \tag{13}$$

Instead, we maximize the log-likelihood ℓ,

$$\ell(\mathbf{\Theta}_M) = \sum_i \sum_t \log Prob(p_i(t)|\mathbf{\Theta}_M, \mathcal{I}_t). \tag{14}$$

Now consider optimization with respect to a particular actors parameters θ_i,

$$\ell(\theta_i) = \sum_t \log Prob(p_i(t)|\mathbf{\Theta}_M, \mathcal{I}_t) +$$
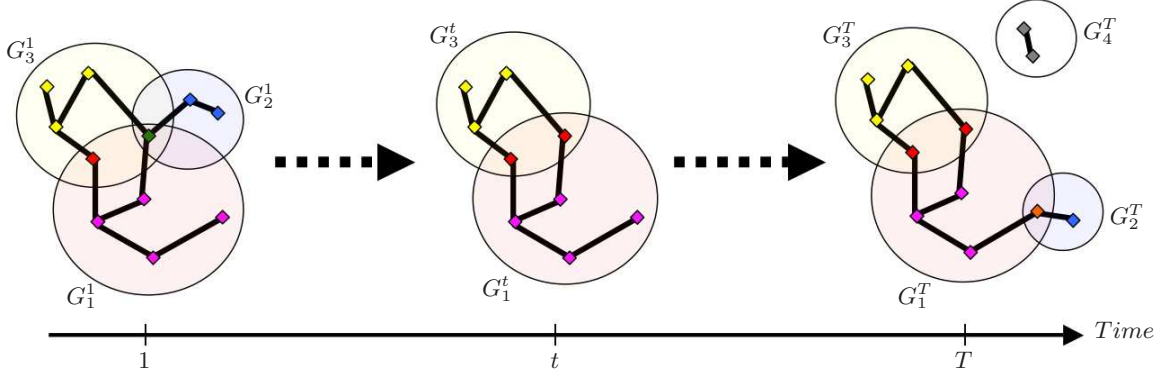$$\sum_{\bar{i} \neq i} \sum_t \log Prob(p_{\bar{i}}(t)|\mathbf{\Theta}_M, \mathcal{I}_t). \tag{15}$$

Figure 3: An example of training data which includes the information about group evolution

The actions of $\bar{i} \neq i$ depends on θ_i only through $\mathcal{I}_t$, which is a second order dependence, therefore we ignore the second term in optimizing the parameters specific to actor i,

$$\theta_i^* \leftarrow \text{argmax} \sum_t \log Prob(p_i(t)|\Theta_M, \mathcal{I}_t). \qquad (16)$$

Thus the maximization over a single actor's parameters only involves that actors 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.

3.1 Example of learning process

We are going to use learning actors' type (§ 2.1.1) to illustrate the learning process. We have a data set which includes the information about group evolution. The group evolution gives us the information about actors' actions (*join a group*, *leave a group*, or *do nothing*) and which groups actors join or leave in each time step. In our model, an actor's type controls which group she most likely joins or leaves (§ 2.3), and how quickly an actor's rank increases in a group (§ 2.2). Therefore, if we have the data of the group evolution, we can use the path of which groups an actor joins or leaves to learn the actor's type. Assume that we know all the values of parameters except the actors' type, then, based on equations (6), (7), and (8), we compute the probability of the path for each actor with different types. So, what is an actor's type is based on which actor's type have the highest probability of the path. The learning algorithm is as follow:

Algorithm 1 Maximum log-likelihood learning algorithm

Input: The set of social groups at each time step, $\mathcal{D} = \{\mathcal{D}_t\}_{t=1}^{T+1}$
Output: The best value for θ_i
1: $x \longleftarrow \{Leader, Socialite, Follower\}$
2: **for** each actor i **do**
3:      find $\mathbf{p}_i^T$, the set of actions, to determine which group actor i joined and left.
4:      **for** each possible value x for θ_i **do**
5:          $Prob_x \longleftarrow \sum_t \log Prob(p_i(t)|\theta_i = x, \Theta_M, \mathcal{I}_t)$
6:      **end for**
7:      $\theta_i^* \longleftarrow \text{argmax}(Prob_x)$
8: **end for**
9: **return** all θ_i^*

Sometimes, we don't know the probability distributions, which are used in equation (6), for actors' group size preference. We develop an expectation-maximization (EM) algorithm to learn the actors' type and also the probability distributions of actors' group size preference. We know different type of actor has different probability distribution of group size preference, and we also know the group size preference influence which group an actor most likely joins or leaves. Based on the group evolution, we are able to find out which size of groups an actor joins and leaves, and we can use this information for all actors with same type to determine the group size preference distribution for that actor's type.

Assume the group size preference for each actor type is a Gaussian distribution, which is different from our model, then we need to determine the means and variances for those Gaussian distributions. We use standard K-means algorithm [12, 4] to cluster actors based on the average size of groups each actor joined into 3 clusters. We, then, compute the mean and the variance of the average group size for each cluster. This is a simple heuristic based on the observation that leaders join small groups, followers large groups, and socialites median groups. The clustering algorithm is as follow:

Algorithm 2 Cluster

1: **for** each actor i **do**
2:      $size_i \longleftarrow$ the average size of groups actor i joined
3: **end for**
4: cluster $\{size_i\}_{i=1}^{|\mathcal{A}|}$ into 3 clusters using the standard 3-means algorithm.
5: **for** each cluster c **do**
6:      $\mu_c \longleftarrow E(size_k)$ where $k \in$ cluster c
7:      $\sigma_c \longleftarrow Var(size_k)$ where $k \in$ cluster c
8: **end for**
9: **return** all μ_c and σ_c

We use the means (μ_c's) and variances (σ_c's) found in *Algorithm 2* as initial means and variances for those Gaussian distributions of group size preference in *Algorithm 1*. After *Algorithm 1*, we are able to obtain a new set of clusters, from which we can calculate the new means and variances. Repeat the whole process to obtain a set of clusters which have maximum probability for all paths of actors' actions. The EM algorithm is as *Algorithm 3*, and we show the results in Section 6.

Algorithm 3 EM algorithm for learning actors' type

Input: The set of social groups at each time step, $\mathcal{D} = \{\mathcal{D}_t\}_{t=1}^{T+1}$
Output: The best value for θ_i
1: compute μ_c's and σ_c's using *Algorithm 2*, Cluster
2: **repeat**
3:     apply μ_c's and σ_c's to *Algorithm 1*
4:     calculate μ_c and σ_c for each *actor type* based on new θ_i^*
5: **until** exceed threshold
6: **return** all θ_i^*

---

4. LEARNING FROM COMMUNICATIONS

The challenge with real data is that the groups structure and their evolution are not known, especially in online communities. Instead, one observes the communication dynamics, *e.g.* Fig. 4 presents an example of training data which only includes the communication dynamics. However, the communication dynamics are indicative of the group dynamics, since a pair of actors who are in many groups together are likely to communicate often. One could one more probabilistic layer on the model linking the group structure to the communications, however, the state space for this hidden Markov model would be prohibitive. We thus opt for a simpler approach. 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.

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 $G_{\tau_1}, G_{\tau_2}, \ldots, G_{\tau_{T+1}}$. The time periods need not be disjoint, and in fact we have used the overlapping time-periods to cluster the communication data into groups as a way of obtaining more stable and more reliable group structure estimates for learning on real data since there is considerable noise in the communications – aggregation, together with the overlap smoothens the time series of communication graphs. This does not affect the probabilistic model approximation of independence among actor paths, or independence among transitions from time-step to time step. 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, one needs to construct the paths of each actor. This means we need the correspondence between groups of time step t and $t+1$, in order to determine the actions of the actors. 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 exist-

ing groups disappeared. In order to find this matching, we use a standard greedy algorithm as follow:

Algorithm 4 Finding matchings

1: 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.
2: We use the symmetric set difference $d(x, y) = 1 - |x \cap y|/|x \cup y|$ as a measure of error between two sets.
3: 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 $d(X_i, Y_j)$.

This problem can be solved in cubic time using max-flow techniques [9]. However, for our purposes, this is too slow, so we use a simple greedy heuristic. First find the best match, i.e. the pair (i^*, j^*) which minimizes $d(X_i, Y_j)$ over all pairs (i, j). This pair is removed from the sets and the process continues. An efficient implementation of this greedy approach can be done in $O(n^2 \log n)$, after $d(X_i, Y_j)$ has been computed for each pair (i, j). The algorithm is shown as follow:

Algorithm 5 Finding matchings using a simple greedy heuristic

---

Input: $\mathcal{X} = \{X_1, \ldots, X_n\}$ and $\mathcal{Y} = \{Y_1, \ldots, Y_n\}$
Output: Pairs of best matching groups.
1: $\mathcal{S} \longleftarrow \emptyset$
2: $\mathcal{M} \longleftarrow \emptyset$
3: **for** $X_i \in \mathcal{X}$ **do**
4:     **for** $Y_j \in \mathcal{Y}$ **do**
5:         compute $d(X_i, Y_j)$
6:         $\mathcal{S} \longleftarrow \mathcal{S} \cup \{i, j, d(X_i, Y_j)\}$
7:     **end for**
8: **end for**
9: sort $\mathcal{S}$ based on $d(X_i, Y_j)$
10: **while** $\mathcal{S} \neq \emptyset$ **do**
11:     find pair (i^*, j^*) which minimizes $d(X_i, Y_j)$
12:     $\mathcal{M} \longleftarrow \mathcal{M} \cup \{i^*, j^*, d(X_{i^*}, Y_{j^*})\}$
13:     $\mathcal{S} \longleftarrow \mathcal{S} - \{k, l, d(X_k, Y_l)\}$ where $k = i^*$ or $l = j^*$
14: **end while**
15: **return** $\mathcal{M}$

---

5. 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 $\mathcal{D}_{test} = \{\mathcal{D}_t\}_{t=T+2}^{T+K}$. 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 (normalized) between the frequencies as well as the squared error difference be-
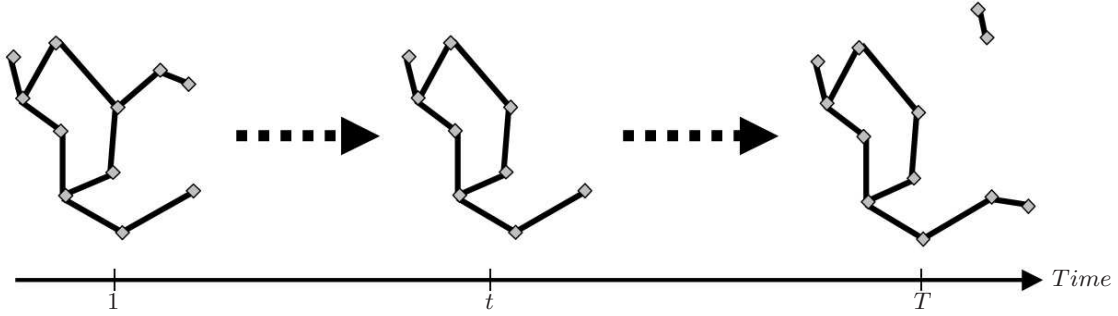
Figure 4: An example of training data which only includes the communication dynamics

tween the probabilities,

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

$$
E_p(t) = \sqrt{\sum_k \left(\frac{N_k^{pred}(t)}{N^{pred}(t)} - \frac{N_k^{true}(t)}{N^{true}(t)}\right)^2},
$$

(17)

where $N^{pred}(t) = \sum_k N_k^{pred}(t)$ and $N^{true}(t) = \sum_k N_k^{true}(t)$. E_p measures the difference in the shape of the histograms, whereas E_f takes into account the actual number of groups.

6. EXPERIMENTS

In our model, the parameters can be learned using the approach described in Section 3 and Section 4. Here we focus on three parameters, the actor's type $type_i$ (see §2.1.1), the initial resource allocation E_i (see §2.1.4), and the society reward/penalty parameters θ_{reward} (see §2.4).

6.1 Learning Actors' Types

In learning actors' type, we evaluate the results from 7 different algorithms:

- Learn: Only use *Algorithm 1* with true distributions for group size preference.

- Cluster: Only use *Algorithm 2* to classify actors' type based on leaders joining small groups, followers large groups, and socialites median groups.

- EM: Use *Algorithm 3* with unknown distributions for group size preference.

- 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.

To evaluate performance, we use an instance of the model to generate simulation data for training and testing. Since we know the model, we can compare the true type with the learned type to compute the accuracy. And we are also able to collect the information about group evolution which avoid pre-processing the training data mentioned in Section 4. We used 50, 100, 150, 200 and 250 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
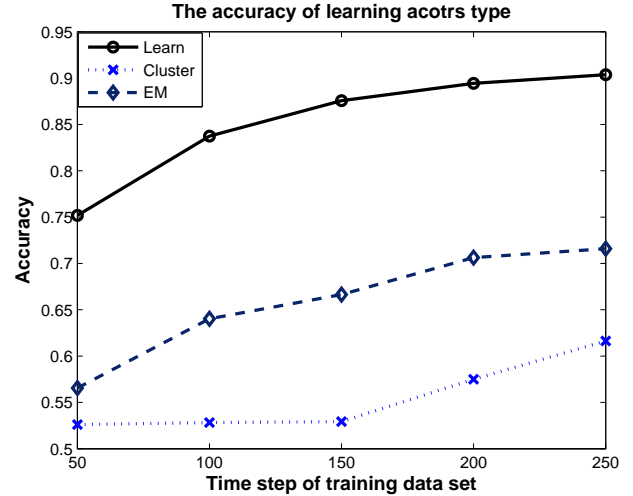

Figure 5: The accuracy of learning actors' type using algorithm Learn, Cluster, and EM

class and simulating for 50, 100, 150, 200 and 250 time steps. All others parameters except types and distributions of group size preference were held fixed.

Figure 5 and Table 1 show the accuracy of Learn, Cluster and EM algorithms with different time steps of training data set, and for comparison, the accuracy of randomly assigning type is 0.33. The results tell that the accuracy for Learn algorithm is the best because it knows the true distribution of group size preference, and only need to learn actors' type. The Cluster algorithm has the worst result, and our EM algorithm does improve the results from the Cluster algorithm, even though we don't know the true distribution of group size preference. In §2.1, we know the type of an actor governs not only the actor's preferences for group size but also her "ambition." The Cluster algorithm is only based on the average size of groups the actor joined which can be detected from group evolution data. On the other hand, the Learn and EM algorithms can also learn actors' ambition, which cannot be observed from the group evolution data. From the results, we also can tell when the length of the time period of training data set increases, we obtain better results from all algorithms. The reason is that more data points we can learn from, more accuracy of the results we can achieve.

Accuracy 61.64%		Learned Actors' Types		
		L	*S*	*F*
True	*L*	68.25	66.75	2.4
Actors'	*S*	0.75	88.5	119.25
Class	*F*	0.0	2.65	151.45

(a) Cluster algorithm

Accuracy 71.59%		Learned Actors' Types		
		L	*S*	*F*
True	*L*	126.05	11.24	0.11
Actors'	*S*	46.55	125.2	36.75
Class	*F*	3.47	43.93	106.7

(b) EM algorithm

Accuracy 90.38%		Learned Actors' Types		
		L	*S*	*F*
True	*L*	137.15	0.25	0.0
Actors'	*S*	20.1	185.75	2.65
Class	*F*	8.3	16.8	129

(c) Learn algorithm

Table 1: Confusion matrix in learning types for Cluster, EM **and** Learn **algorithms. The accuracies are (a) 61.64%, (b) 71.59%, and (c) 90.38% on 250 time steps of training data.**

The multi-step predictive performance of the algorithms on test data according to the metrics in equation (17) is shown in Fig. 6, from which it is clear that Learn surpasses all the algorithms because the prediction from Learn is most close to Optimal (which is unattainable in reality), and the worst cases are Leader and Follower which assign all actors to leader or follower types.

In a similar fashion, one could learn E_i and θ_{reward}, but we do not give detailed results here. Instead, in the following section, we consider using the model to determine which parameters are worth learning and which are not based on predictive performance.

6.2 Impact of Learning on Prediction

The learning process is time consuming; for instance, in a community, there are N actors and K groups. Then, in each time step, there are $(2^{(NK)}/K!)$ possible actors' combinations to establish groups. If we have data for T time steps, the complexity of finding the optimal path using dynamic programming is $O(T \times \frac{2^{NK}}{K!}) \approx O(T \times 2^{K(N-logK)})$. Even though, we are able to develop some approximate and efficient algorithms, but when having this complexity, the learning process is still very time consuming. Therefore, it is better to know if the parameter has significantly improved the predictive performance before the learning takes place.

From Fig. 6, it is clear that the types of actors significantly impact the predictive performance as compares to other learning algorithms. While we could learn $\{E_i\}$ and θ_{reward}, perhaps it is not useful to learn them if they don't significantly improve predictive performance. Figure 7 shows the impact of the optimal predictor (all parameters set to their true values) vs. choosing a random θ_{reward}, and for various choices of $\{E_i\}$ (every actor assigned some fixed E according to max$\{E_i\}$, min$\{E_i\}$ or average$\{E_i\}$). We compute the average errors from 20 runs. As can be seen, the wrong θ_{reward} does significantly affect the performance, but using the wrong $\{E_i\}$ does not (which results mixed with the optimal predictor have the average error around 0.29 in (a) and 0.08 in (b))

– which might be expected as the effects of initial conditions should equilibrate out.

6.3 Results on Real Data

Our results on real data are based on communications because it is difficult to collect data that includes the group evolution from the real world. Hence, we need to use the algorithms in Section 4 to obtain the group evolution from communication dynamics – using the overlapping clustering algorithm [3] to obtain the groups information in each time step, and then apply the matching algorithm, *Algorithm 5*, to find out the group evolution. After we have the data about the group evolution, then apply the technique mentioned in Section 3 to learn the parameters.

We collected the communication data from a movie newsgroup, which includes 1528 active actors in 206 days. We use each 2 days as one time step, and have total 103 time steps. Here, we are going to show the process of learning actors' type. We use the algorithms in Section 4 to obtain the group evolution, and then apply the learning algorithms, *Algorithm 2* or *Algorithm 3* in Section 3 to learn the appropriate values of actors' type. The results in previous section indicate that one can learn types and improve predictive performance. We apply both EM and Cluster algorithms on the real data set, and the results are shown in Table 2. The results are interested because based on Cluster algorithms, the majority of actors are leaders which *only* mean they prefer joining small groups, a finding that was made by hand on newsgroups data through social analysis by Butler [6]. It makes sense because it is easy to observe which size of groups an actor is most likely to join by hand; however, it is difficult to determine how an actor's ambition in its activities by hand. Accounting to the results of EM algorithm, the number of *Follower* increases 30.9%, the number of *Leader* decreases 19%, and the number of *Socialite* decreases 11.9%. The results discover the behavior that a lot of actors are not so ambitious. It is also make sense because in a newsgroup community, the majority of actors would more like to read news than post news. Our algorithms additionally identify which actors are of which type, group size preference and ambition, and are fully automated.

	Learned Actors' Types		
	Leader	*Socialite*	*Follower*
Number of Actor	822	550	156
Percentage	53.8%	36.0%	10.2%

(a) Cluster algorithm

|	Learned Actors' Types		
	Leader	*Socialite*	*Follower*
Number of Actor	532	368	628
Percentage	34.8%	24.1%	41.1%

(b) EM algorithm

Table 2: The result of learning actors' type on a movie newsgroup using EM **and** Cluster **algorithms.**

7. DISCUSSION

We have presented a parameterized stochastic model for learning the micro-laws governing a society's social group dynamics. Parameterized stochastic models in machine learning are not uncommon, however in the context of learning social laws, they have scarcely been applied. Our main contributions are the application of efficient algorithms and heuristics toward learning the parameters in the specific application of modeling social groups. Our re-
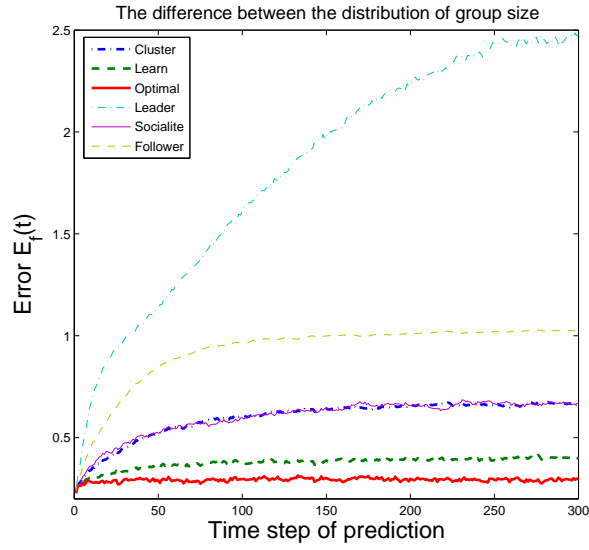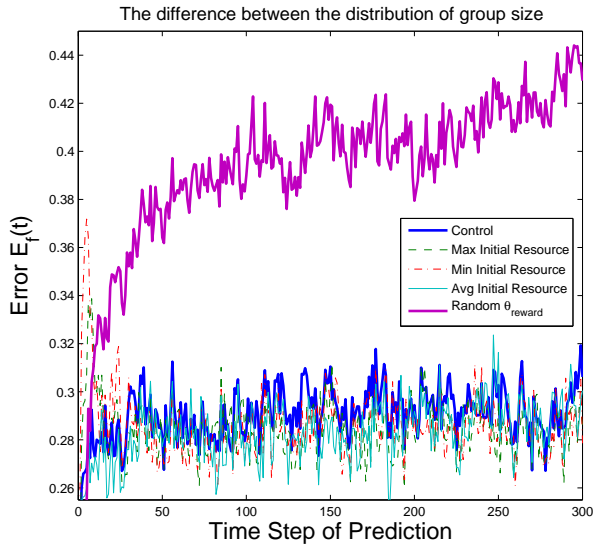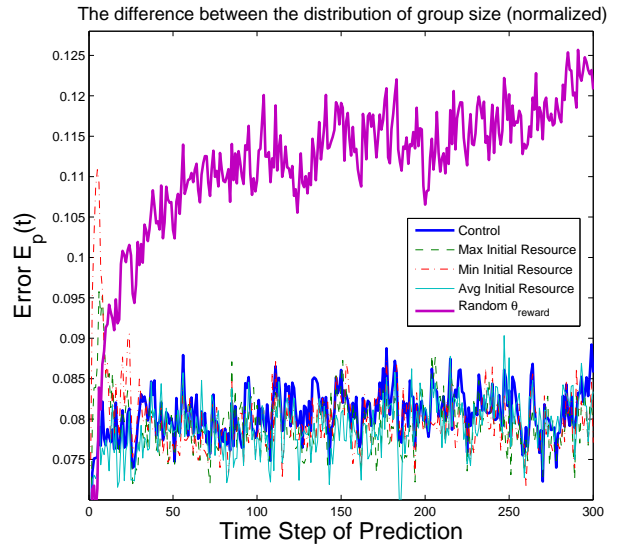
(a) E_f
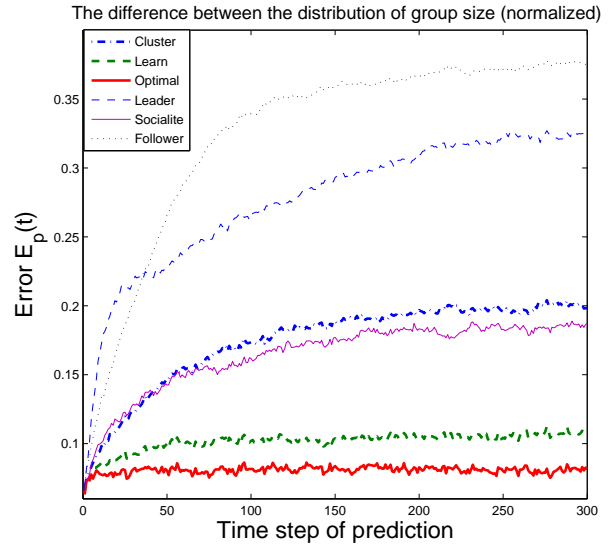
(b) E_p

Figure 6: The predictive error for various algorithms to learn type.

(a) E_f

(b) E_p

Figure 7: Impact of using incorrect θ_{reward} and $\{E_i\}$ on predictive error.

sults on simulated data indicate that when the model is well specified, the learning is accurate. Since the model is sufficiently general and grounded in social science theory, some instance of the model should be appropriate for any given society. In this sense almost any general model of this form which is founded in social science theory should yield similar results.

Within our model, one of the interesting conclusions from a social science point of view is the ability to identify which parameters have a significant impact on the future evolution of the society. In particular, the initial resource allocation (a fixed intellectual resource of the actor) does not seem to be a dominant factor, but the actor ambition and size preference, as well as the penalty/reward structure for the society have significant impact on the evolution. These types of conclusions are exactly the types of information a social scientist is trying to discern from observable data. Our results are not completely surprising. A community which excessively penalizes traitors (say by sever dismemberment), does necessarily see far fewer changes in its social group structure than one that doesn't. Our observations indicate that learning effort should be focused on certain parameters and perhaps not as importantly on others, which can further enhance the efficiency of the learning. Having determined what the important parameters are, one can then go ahead and learn their values for specific communities.

8. ACKNOWLEDGMENTS

9. 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] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

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

[6] B. Butler. The dynamics of cyberspace: Examining and modeling online social structure. Technical report, Carnegie Melon University, Pittsburgh, PA, 1999.

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

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

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. Mcgraw-Hill, Cambridge, MA, 2nd edition, 2001.

[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] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.

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

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

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

[16] R. D. Putnam. *Bowling alone : the collapse and revival of American community*. Simon and Schuster, 2000.

[17] 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.

[18] T. Snijders. The statistical evaluation of social network dynamics. In M. Sobel and M. Becker, editors, *Sociological Methodology Dynamics*, pages 361–395. Basil Blackwell, Boston & London, 2001.

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