Graphical Multiagent Models

by

Quang A. Duong

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2012

Doctoral Committee:

Professor Michael P. Wellman, Chair
Professor Satinder Singh Baveja
Professor Edmund H. Durfee
Assistant Professor Long Nguyen

To Mom, Dad, and Mai.

ACKNOWLEDGEMENTS

I am immensely grateful to my family, friends, and colleagues for their indispensable support and encouragement while writing this dissertation. First of all, I would like to thank my advisor, Michael Wellman, very much for inspiring and guiding me through the highs and lows of the doctorate program. His tremendous patience and encouragement have allowed me to explore many different research ideas, making the last five years my most productive and exciting time yet. It is a true honor and a great pleasure for me to work with such an inspiring scientist like Michael on this long journey.

I would not be able to complete my dissertation without invaluable reviews and helpful comments and suggestions from my committee members: Satinder Singh Baveja, Edmund Durfee, and Long Nguyen. I also want to give special thanks to Satinder for his vital guidance during my dissertation research work.

Many professors and instructors have inspired my academic voyage over the years. I am profoundly indebted to my high school teacher, Nguyen Thanh Hung (Hung Nguyen), who introduced me to the fascinating world of computing and instilled me with a passion for programming. I am deeply thankful to my two college advisors, Gordon Beavers at the University of Arkansas and David Parkes at Harvard University, for helping me realize my potentials, and showing me a world of many possibilities.

I have been very fortunate to work with and learn from a number of phenomenal colleagues from various organizations. I would like to thank Yevgeniy "Eugene" Vorobeychik for various fruitful collaborations on the early development of my dissertation research, Michael Kearns for the use of human-subject data from experiments conducted by his re-

search group, and Gregory Frazier for having been both an excellent collaborator and an encouraging and patient supporter over the past three years. My time spent as a research intern at Yahoo! Research during the summer of 2011 is hugely influential to the shaping of my professional career. I am tremendously grateful to have such an incredible level of intellectual freedom and intensity when working with many outstanding researchers at Yahoo! Research, such as David Pennock, Sebastien Lahaie, Jake Hofman, Sharad Goel, and Sergei Vassilvitskii.

I have many remarkable lab mates, whose insights and support are critical to my achievements. Many thanks to Patrick Jordan for motivating me with his exemplary work ethics, and for giving me insightful research and career advice. Matthew Burgess, Lee Callender, Ben Cassell, Yagil Engel, James "Augie" Hill, Akshat Kaul, Christopher Kiekintveld, Chao-Lin Liu, Bartley Tablante, Prateek Tandon, Sri Krishna Vempati, Elaine Wah, Bryce Wiedenbeck, and Dong Young Yoon are all invaluable colleagues and friends.

I am incredibly lucky to be able to spend the last five years with many amazing and supportive friends: James Boerkoel, Jeshua Bratman, Robert Cohn, Gargi Datta, Nathaniel Derbinsky, Laura Fernandez, Naseem Jauhar, Miku Kawakami, Alexander Kuhn, Ryan Morton, Azarias Reda, Johannes Strom, Nabihah Tayob, and many others. It has been a great pleasure for me to share my graduate school experience with Daniel Fabbri, an extraodinary colleague, housemate, and friend, whose companionship was vital through trying times. I would like to especially thank my housemates over the years, Jose "Hiram" Avalos, Casey Herron, Steven Hoffenson, Ashley King, Erin Payne, Alexandra Turner, and Jordan Zastrow, for always being there for me and cheering me on through all the ups and downs.

I started my journey in Ann Arbor with a warm welcome by my college friends, Oanh Nguyen and Dang Vu, who since then have done so much to provide me with a home away from home. I cannot thank Marie "Chau" Will and Nathan Will enough for believing in me, and genuinely caring for me just like family, even from hundreds of miles away.

Above all, I would like to express my utmost gratitude to my mom, Nguyen Thi Ngoc Thanh (Thanh Nguyen), my dad, Duong Ngoc Hai (Hai Duong), and my sister, Duong Thanh Mai (Mai Duong), for their unconditional love and support.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

ABSTRACT

Graphical Multiagent Models

by

Quang A. Duong

Chair: Michael P. Wellman

I introduce *Graphical Multiagent Models* (GMMs): probabilistic graphical models that capture agent interactions in factored representations for efficient inference about agent behaviors. The graphical model formalism exploits locality of agent interactions to support compact expression, and provides a repertoire of inference algorithms for efficient reasoning about joint behavior. I demonstrate that the GMM representation is sufficiently flexible to accommodate diverse sources of knowledge about agent behavior, and to combine these for improved predictions. *History-dependent GMMs* (hGMMs) extend the static form of GMMs to support representation and reasoning about multiagent behavior over time, offering the particular advantage of capturing joint dependencies induced by abstraction of history information. Computational experiments demonstrate the benefits of explicitly modeling joint behavior and allowing expression of action dependencies when given limited history information, compared to alternative models.

Many multiagent modeling tasks that employ probabilistic models entail constructing dependence graph structures from observational data. In the context of graphical games where agents' payoffs depend only on actions of agents in their local neighborhoods, I for-

mally describe the problem of learning payoff dependence structures from limited payoff observations, and investigate several learning algorithms based on minimizing empirical loss. I show that similar algorithms can also be applied to learning both dependence structures and parameters for hGMMs from time-series data. I employ data on human-subject experiments in constructing hGMMs and evaluating the learned models' prediction performance for a consensus dynamics scenario. Analysis of learned graphical structures reveals patterns of action dependence not directly reflected in observed interactions. The problem of modeling information diffusion on partially observed networks provides another demonstration of hGMM flexibility, as unobserved edges may induce correlations among node states. As graphical multiagent models can compensate for correlations induced from missing edges, I find that learning graphical multiagent models for a given network structure from diffusion traces can outperform directly recovering the missing edges, across various network settings.

CHAPTER I

Introduction

1.1 Motivation

Consider an example multiagent scenario where people (agents) in a small town named AA need to pick their mayor in a democratic election that pits two candidates, one Democratic (blue) and one Republican (red), against each other. Voters are often engaged with their friends, family, and colleagues in lively political conversations with the intention to persuade others to agree with their present candidate pick, and eventually choose the same candidate. Each town-person is free to choose which candidate to support; at the same time, his or her decision is also affected by interactions with other voters within his or her own social circle. Furthermore, as people in the small town of AA greatly value community unity and despise political inaction and stalemate prevalent in their nation, they seek to avoid divisive election outcomes and prefer voting results as close to consensus as possible. They also openly discuss their final votes after the election.

Given data about voters' party preferences in past elections and previous public opinion polls, a political analyst (system modeler) named May would like to make predictions about the upcoming election. In addition, she presumably has some estimate of the town's relationship graph, which essentially specifies each person's social circle and effectively who talks to whom in town. In particular, May is interested in predicting the election's outcome, and what percentage of the vote the winning party will receive. She may also have

a narrower focus on predicting which candidate specific key community leaders will likely support. In order to complete these prediction tasks, she constructs a multiagent model that captures both agent reasoning about the pros and cons of each voting option and the potential influences exerted by other agents in the system. This model basically defines a joint probability distribution over possible election outcomes, taking into account beliefs about agent reasoning and their interactions. May can then use the model to answer various inference queries about the election's outcomes. For instance, the model's probability distribution over all possible outcomes essentially indicates each party's winning chance; the marginal probability distribution of a subset of community leaders helps to infer about the leaders' support; the conditional probability distribution of a community or a district's candidate picks given their leaders' picks implies how much one can affect election outcomes by winning community leaders' endorsements.

Activities in our society usually involve multiple *autonomous* organizations and persons that either directly or indirectly *interact* within their shared environments, together composing different types of *multiagent systems*, the main study object of the thesis. The rise in complexity of human social interactions in large systems has raised the need for understanding and predicting the causes and consequences of such interactions, while presenting great challenges for designing models that can compactly and effectively represent these systems' dynamics. For instance, the ability to analyze and predict the contagion of systemic risks among interconnected financial institutions pursuing different strategies bears important implications on the drafting process of financial regulations. A good understanding of how people influence each other's preferences and actions in online social networks not only contributes to the advancement of viral marketing but also provides insights on what political campaigns can do to reach and expand their support bases. Similarly, a model of interactions among many different networked (autonomous) computers and servers will help network system designers understand the effectiveness of new routing schemes or network configurations in discouraging malicious software activities and

preventing computer virus spreads.

Consider the problem of modeling a large and complex multiagent system, such as a market, local election, social network, or computer network. The main goal here is to model agent behavior as their interactions play out over time, in a form that can be efficiently represented and employed in prediction or other uncertain reasoning tasks. One may base the model on some general theory of agent behavior, as for instance in the increasingly popular game-theoretic approach which treats agents as perfectly rational decision makers. This is a helpful starting point in many situations, but falls short to the extent that the general theory fails to account for multiagent behaviors beyond the theory's assumptions. In the example of game-theoretic modeling, the modeler might have evidence that agents may deviate from rational behavior. Moreover, the modeler may have alternative or complementary theories, or statistical evidence that could inform the modeling exercise. I therefore propose a framework that is agnostic about fundamental drivers of agent behavior, but allows any assumptions or available knowledge about such drivers to be incorporated in an overall probabilistic characterization of multiagent activity.

1.2 Challenges and Approaches

The enormous space of joint behaviors for a large multiagent system renders general representations intractable. For instance, the modeler of the aforementioned town mayor election example would need to specify the probability distribution function over all possible full outcomes, each of which specifies every person's vote. This is a very challenging task, especially given that the total number of full outcomes is exponential in the town's population. Computationally feasible approaches must therefore exploit some structure in these behaviors, for example, locality of interaction among agents. Probabilistic graphical models have proved useful for capturing localized effects in a variety of contexts, and so are a logical starting point for the design of compact and effective multiagent representations. This idea has been the driving motivation for several representations of large-scale

game models, such as multiagent influence diagrams (MAIDs) (Koller and Milch, 2003), action-graph games (Jiang et al., 2011) and graphical games (Kearns et al., 2001). In the basic graphical game approach, for instance, the model is a factored representation of a normal-form game, and special-purpose algorithms operate on this representation to identify approximate or exact Nash equilibria. Whereas a game model directly represents the decision facing agents rather than their actual behavior, a graphical model can capture behavioral dependence patterns manifested in the locality of agents' decision interactions. Kakade et al. (2003) and Daskalakis and Papadimitriou (2006) also draw computational connections between game and probabilistic graphical models, by demonstrating how to map a graphical game to a Markov random field for the purpose of exploiting probabilistic inference methods.

The modeler faces even larger challenges when seeking to capture multiagent behaviors in dynamic settings. Multiagent research has produced many forms of models for dynamic systems, taking into account a broad range of factors. Generative models (e.g., based on rules for individual agent behavior) are highly complex, offering virtually unlimited capacity for specifying any dynamic behavior the modeler may conceive. Reasoning about the properties of such models (e.g., for prediction based on partial observation), however, is often challenging computationally, requiring in general a prohibitive amount of sampling of the possible system trajectories. Analytic models, in contrast, may facilitate more efficient reasoning, but often at the cost of strong limits on the complexity of the agent behavior and environment that can be represented.

One source of complexity in multiagent modeling is the interdependence of agent behaviors due to interactions between agents. Even if agents are autonomous (i.e., make decisions independently), conditioning their decisions on common observations (including each others' actions) may induce behavior correlations over time. Generative models typically specify agent behaviors individually, exploiting the conditional independence of their actions given system history. However, the modeler is often interested in properties of

system behavior when given only an incomplete view of this history, or forced to limit how much of this history can be incorporated in the model due to computational constraints. In these cases, agent decisions are not independent, and one may wish to directly express joint behavior. For instance, it is reasonable to assume that after all their heated political discussions, most town people in AA themselves eventually decide who to vote for on election day. In this scenario, the modeler May seeks to predict the election outcome given opinion polls (i.e., history) collected prior to election day. However, budget constraints permit her to conduct only a small number of polls (i.e., limited history) for her analyses, challenging her ability to provide reliable predictions about the election. Furthermore, she can incorporate in the model only some statistics of the opinion polls, such as what percentage of a community declared Democratic in the most recent poll or how many percentage points one candidate gained in the last three polls, instead of data on every single voter's currently favored candidate.

In many scenarios the modeler has noisy or limited information regarding the graphical structure underlying agent interactions in the system. As factored models of multiagent systems are built upon the graphical structures specifying locality in agent interactions, discovering these interaction structures is imperative to multiagent behavior modeling tasks. Moreover, agents often interact simultaneously through different channels, such as person-to-person, emailing and instant messaging, at varying levels of engagement, from heated political debates to merely observations of others' actions. As a result, it is non-trivial which interactions should be included in the final model, even when the modeler knows exactly all the existing connections and their characteristics. In fact, the graphical structure component of a probabilistic model, which captures probabilistic relations between connected nodes (agents), is conceptually different from the graph of agent interactions often given as input. For example, the town's relationship graph that May employs in her modeling effort might accurately reflect family and work colleague ties based on official census data and employment records, but may poorly capture other relationships, such as friend-

ships. In settings of stricter privacy policies, one may even imagine that the modeler has no access to detailed employment and household records, making any relationship graph inaccessible. Moreover, even though May knows that A and B are coworkers and thus connected in the relationship graph, she may still not be certain if A listens to or respects B's opinions and vice versa, or in other words, whether their voting behaviors are potentially correlated.

1.3 Propositions

To address the problem of modeling multiagent behavior, I introduce *graphical multiagent models* (GMMs) (Duong et al., 2008) and *history-dependent graphical multiagent models* (hGMMs) (Duong et al., 2010, 2012). They are basically probabilistic graphical models that allow direct expression of different behavior dependencies and correlations, and the joint probability distribution of agent actions. In the body of this thesis, I investigate the representational and predictive power of GMMs and hGMMs, as well as their contributions to the advancement and application of graphical models in analyzing and understanding real-world large complex multiagent systems. The rest of this chapter summarizes the propositions addressed and studied in my thesis.

1.3.1 GMMs Facilitate Expression of Beliefs about Agents' Actions Derived from a Variety of Sources

In a graphical multiagent model, the joint probability distribution is interpreted as an uncertain belief (e.g., a prediction) about the agents' play (joint action). The GMM framework described and discussed in Chapter III allows that beliefs about agent actions may be based on various solution concepts, models of equilibrium selection, or for that matter any alternative theory or knowledge about agent reasoning. The Daskalakis and Papadimitriou (2006) mapping from pure strategy Nash equilibria (PSNE) to maximum a posterior estimates of graphical models, as described in Section 3.2, can be viewed as a GMM where

it is believed that the agents will play a PSNE if one exists. Similarly, the Markov network formalism of graphical games (Kakade et al., 2003) is a form of GMM, in this case representing the set of correlated equilibria. However, game-theoretic analysis is only one source of beliefs for multiagent behavior. Even when the scenario of interest is described as a game, one may wish to adopt alternative bases for forming beliefs about the agents' actions (Vorobeychik and Wellman, 2006). At this level, my motivation shares the spirit of the network of influence diagrams formalism of Gal and Pfeffer (2008), which extends MAIDs to incorporate non-maximizing models of behavior. It also aligns with the goal of Wolpert's information-theoretic framework for modeling bounded rationality in game play (Wolpert, 2006).

The ability to accommodate different beliefs about agents' actions further allows the modeler to combine GMMs from diverse knowledge sources in a unified model with improved prediction power, which is illustrated and analyzed in Sections 3.3 and 3.4. In other words, one can exploit this flexibility in representation by employing multiple knowledge sources to improve prediction outcomes, which I demonstrate for the task of predicting the outcome of a reinforcement learning process.

1.3.2 By Augmenting GMMs with Across-Time Dependencies, History-Dependent GMM Can Capture Dynamic Multiagent Scenarios

There are usually more sources for action dependence among agents in dynamic settings than in static settings. As a result, the ability to express joint action dependencies directly in history-dependent GMMs, introduced and defined in Chapter IV, is increasingly more important for modeling dynamic scenarios. For example, Section 4.2 illustrates that the incorporation of full history in analytical models is often computationally expensive, subsequently opening the door for more action correlations as abstracted or summarized history is used instead. In the empirical study of Section 4.4 on a consensus dynamics scenario, I demonstrate that even in settings where agents act autonomously, joint behav-

ior history-dependent GMMs can still provide better prediction performance than models that do not allow joint behavior specification. Through empirical studies using simulated as well as human-subject data, I show that history-dependent GMMs learned from data can produce predictive performance exceeding hand-crafted models. At the same time, history-dependent GMMs can exploit localized effects of interactions, thus supporting efficient inference about system dynamics based on abstracted information about prior actions.

1.3.3 It Is Feasible to Learn Both Dependence Graph Structure and Model Parameters from Time-Series Data

The study presented in Chapter V illustrates the difference between agent interaction graphs and dependence graphs, emphasizing the need for learning dependence graphs from observational data sources and employing them in inference tasks. In particular, I first formulate and address the problem of learning compact representations of a multiagent interaction manifested through payoff interdependence in Section 5.1. Based on that study's findings on the effectiveness and efficacy of different structure learning algorithms, I then introduce a greedy algorithm to learn history-dependent GMM dependence graph structures capturing probabilistic correlations among agent actions, as well as the model parameters, as described in Section 5.2. Analysis of learned history-dependent GMM structures, presented in Section 5.2.3, further elucidates the distinction between dependencies that are pivotal for representation and those suggested by observed inter-agent interactions in a dynamic networked setting. This distinction can potentially contribute to the push for richer and more informative representations of diverse relationships among agents in modeling networked behavior.

1.3.4 Modeling Joint Behavior Can Potentially Compensate for Missing or Unobservable Dependence Graph Edges

Even when given a dependence structure as input, the modeler often faces the problem of missing or unobservable edges in the given graph. In Chapter VI, I demonstrate that history-dependent graphical multiagent models allow one to model dynamic behavior without having to reconstruct missing edges using observation data in the problem domain of modeling information diffusion on social networks. In particular, Section 6.3 presents the argument that missing or unobservable edges in this information diffusion scenario induces dependencies that can be captured in history-dependent GMMs, but not the standard cascade models. Note that the illustrating network scenario that I employ is itself an addenda and extension to the repertoire of previously studied systems, as there exists no explicit definition of agent payoff or utility in modeling information diffusion. The empirical study in Section 6.4 illustrates that modeling joint behavior using history-dependent GMMs produces better predictions of information diffusion than alternative approaches that directly seek to reconstruct missing edges while limiting explicit joint behavior expressions.

1.4 Thesis Overview

In Chapter II, I introduce definitions and notation used throughout the thesis, and review different approaches to modeling multiagent behavior that form the basis for my probabilistic graphical model approach. Chapter III provides a formal description of GMMs in static scenarios, and discusses their advantages with illustrative examples. Furthermore, I introduce an example scenario of strategic decisions among interconnected technology companies, and empirically demonstrate the benefits of combining different knowledge sources of multiagent behavior into a unified GMM. Chapter IV extends the static GMM representation to dynamic multiagent scenarios, by conditioning joint behavior on abstracted history. I focus on a multiagent scenario, consensus dynamics, which is similar to the afore-

mentioned town election example, and provide motivations and empirical evidence for the effectiveness of history-dependent GMMs in dynamic situations. I proceed in Chapter V to demonstrate how one can learn and predict dynamic multiagent behavior on a network, particularly focusing on learning the dependence graph associated with the same scenario of consensus dynamics introduced and studied in Chapter IV. In Chapter VI, I introduce solutions to the problem of modeling information diffusion in networks with unobserved edges, and subsequently show that the predictive performance of history-dependent GMMs is robust to missing edges in various settings of information diffusion with a considerable number of missing edges. Chapter VII concludes the thesis with a summary of contributions, insights, future research directions, and potential applications.

CHAPTER II

Modeling Behavior in Multiagent Systems

Multiagent system research has encompassed many approaches to modeling agent behavior. Factors based on social norms, preferences, motives, conventions, authority structures, beliefs about others, and more have been incorporated in various multiagent models. Different modeling approaches can be characterized to some extent based on the factors they emphasize, and their underlying assumptions about agent capabilities and tendencies. For example, *game-theoretic models* fully specify agents' beliefs and preferences, and assume agents act rationally according to these attitudes. Within the game-theoretic framework, some modelers postulate adaptive learning processes that can lead agents to adopt game-theoretic solutions (Fudenberg and Levine, 1998). Others may employ dynamic models based on reinforcement learning, irrespective of game-theoretic interests, or adopt behaviors generated by alternative dynamics, for example based on evolutionary models (Tuyls and Parsons, 2007) or combinations (Hennes et al., 2009).

A large number of models focus on particular decision contexts (e.g., social networks, organizations, markets, and teams), and develop behavior rules that are justified by normative or descriptive theories for those contexts. For instance, models of various diffusion processes, such as disease spreads, political movements, or product adoptions, are based on theories ranging from epidemiology and sociology to economics and physics, and largely fall into two different classes. *Cascade* models (Kempe et al., 2003; Goldenberg et al.,

2001) posit that an agent's tendency to be infected, which entails receiving and retaining some spreading information, increases with the proportion of its infected *neighbors*, with whom the agent is in direct contact or *connected*. In *threshold* models (Granovetter, 1978), an agent becomes infected when the number of its infected neighbors exceeds some predetermined threshold.

Ultimately any of the aforementioned models, from different modeling approaches, defines a system, where agent strategies or uncertain environment factors induce a probability distribution over behaviors of the multiagent system. Note that I do not intend to survey the full range of multiagent modeling approaches, much less to comparatively evaluate them. Rather, I seek a representational framework that can accommodate a broad class of behavioral models, and support inference and learning tasks based on such models. In this chapter, I introduce notation and background knowledge necessary for describing multiagent systems and their behaviors in both static and dynamic settings. I further present related foundational work upon which my multiagent system modeling framework is built.

2.1 Multiagent Systems

Consider a multiagent scenario with n agents, where each agent $i \in \{1, \ldots, n\}$ chooses an action a_i, from action domain, A_i. The action choice or *joint action* of all agents in the system is denoted by $a = (a_1, \ldots, a_n)$. Let a_{-i} denote the action choice of all agents except for i. In static settings, agents choose their actions only once. Agents in dynamic settings can change their actions over a time period of length T: at time $t \in [0, T]$, agent i's action is denoted as a_i^t. I focus here on scenarios where the only dynamic elements are agent actions, which allows specifying the system's history at time t as $H^t = (a^0, \ldots, a^t)$. As the capacity of agents' memory is often limited, H^t includes information only for some finite time *horizon* h: $H^t = \{(a_1^k, \ldots, a_n^k) \mid k \in [t-h, t)\}$. To allow that an agent may ignore some past actions of others, I denote by H_i^t the subset of history relevant to i's choice of next action.

2.2 Game-Theoretic Solution Concepts

As the modeling framework proposed in the thesis expands on existing game-theoretic models for decomposable multiagent scenarios, I next provide an overview of basic game-theory notation and concepts.

I refer to agents in game scenarios as *players*, and the actions they choose as *strategies*. Consider a one-shot *game* of n players. In this game setting, each player or agent i chooses a *pure strategy* $a_i \in A_i$. The *strategy profile* $a = (a_i, \ldots, a_n)$ specifies the strategies of all players. Player i then receives a reward or utility $u_i(a) : \prod_i A_i \rightarrow \mathbb{R}$, which depends on its own strategy, as well as other players' strategies in the game.

A *mixed strategy* p_i of player i is essentially an assignment of probability to each pure strategy $a_i \in A_i$ such that $\sum_{a_i \in A_i} p_i(a_i) = 1$, allowing player i to stochastically select strategies. The concept of mixed strategy also leads to the definition of i's expected reward as $u_i(p) = \sum_{a \in A}[\prod_j p_j(a_j)]u_i(a)$. In multi-round games, one may further define agent i's utility over the system's entire time span T as a function of the system's full history over time T: $u_i(H^T)$.

Definition II.1. Best Response

Player i *best responds* to other players' actions a_{-i} if and only if its (pure) strategy a_i satisfies:

$$a_i = \underset{a'_i \in A_i}{\operatorname{argmax}}\, u_i(a'_i, a_{-i}).$$

Any mixture of pure-strategy best responses is a mixed-strategy best response.

Definition II.2. Pure Strategy Nash Equilibrium (PSNE)

A strategy profile a is a pure strategy Nash equilibrium if and only if every agent i plays a best response pure strategy a_i to other players' a_{-i}:

$$\forall i : u_i(a_i, a_{-i}) \geq u_i(a'_i, a_{-i})$$

Similarly, in a mixed strategy Nash equilibrium, no unilateral deviation by any player is profitable to that player.

A related general game-theoretic solution concept is *correlated equilibrium*, which assumes that players have some randomization device they may all observe simultaneously. In general, let $p(a)$ represent the joint distribution over $a \in A$ ($p(a) \geq 0$ and $\sum p(a) = 1$). Also, let a, a joint strategy profile drawn from a joint distribution $p(a)$, be the randomization device's signal. Each player then separately and privately observes its own assigned strategy a_i in a. As in the Nash equilibrium, the distribution p on A is a correlated equilibrium if each player i has no incentive to choose a different strategy a'_i from its assigned strategy a_i, all else equal.

Definition II.3. Correlated Equilibrium

A correlated equilibrium is a distribution p on A such that for all players i and all strategies $a_i, a'_i \in A_i$, conditioned on a_i being in the profile a drawn from p, the expected reward for player i from playing a_i is no smaller than playing a'_i.

It follows that a mixed strategy Nash equilibrium is a correlated equilibrium that happens to be a product distribution, that is $p(a) = \prod_i p_i(a_i)$.

Definition II.4. Regret

For a given payoff model, i's *regret* $\epsilon_i(a)$ represents the maximum gain i can obtain through unilaterally changing its own strategy a_i given a_{-i},

$$\epsilon_i(a) = \max_{a'_i \in A_i} u_i(a'_i, a_{-i}) - u_i(a).$$

It follows that players all have zero regret when playing a Nash equilibrium.

2.3 Graphical Representations of Multiagent Systems

Direct representation of multiagent behavior may require space exponential in the number of agents, which prompts me to seek structures that enable more compact specification. Multiagent scenarios may be particularly amenable to decomposition, to the extent that interactions among the agents exhibit localized effects. Graphical models provide a compact representation for domains with decomposable structure, with concomitant computational advantages. This observation has led to numerous models that exploit locality of agent interaction, including: multiagent influence diagrams (MAIDs) (Koller and Milch, 2003), networks of influence diagrams (Gal and Pfeffer, 2008), and interactive influence diagrams (Zeng et al., 2007). This observation was also a driving motivation for *graphical game* models, first introduced by Kearns et al. (2001), and subsequently examined and extended in several research efforts (Kearns, 2007).

In a graphical game model, a link (directed or undirected) between two players represents a payoff dependence between them. Formally, a graphical game model is a tuple $\mathcal{G} = (I, \{A_i\}, \{J_i\}, \{u_i(\cdot)\})$, where $I = \{1, \ldots, n\}$ is the set of players, and A_i denotes player i's strategy space. J_i is a collection of players connected to i, and $u_i(a_i, a_{J_i})$ is the payoff to player i playing $a_i \in A_i$ when the players J_i jointly play a_{J_i}. The notation $a_{J_i} \subset a_{-i}$ means that each $j \in J_i$ plays its assigned strategy from a_{-i}. Let A_{J_i} be the set of joint strategies of players in i's neighborhood. This game structure is captured by a graph (V, E) in which $V = I$ (i.e., each node corresponds to a player) and there is an edge $e = (j, i) \in E$ if and only if $j \in J_i$ (directed edge) or $j \in J_i \wedge i \in J_j$ (undirected).

The graphical game model is basically a factored representation of a normal-form game, and special-purpose algorithms can operate on this representation to identify approximate or exact Nash equilibria. Kakade et al. (2003) introduced a mapping of joint distributions over strategies in graphical games to a the joint probability distribution of a Markov network or Markov random field (MRF) (Kindermann and Shell, 1980). Daskalakis and Papadimitriou (2006) outlined a special case of this mapping by assigning high potential to

configurations where an agent plays a best response to its neighbors, and showed that the maximum a posteriori configurations of the MRF correspond to PSNE of the game. Their approach enables the exploitation of statistical inference tools for game-theoretic computation, including the full repertoire of graphical model algorithms.

Graphical game models can take advantage of independence only among agents who never interact, between whom no edge exists in the corresponding graphical game structure. Game scenarios that lack this kind of structure, which I label *agent-centric*, can still be compactly represented under the framework of action-graph games (Jiang et al., 2011). Action-graph games explicitly exploit independence in action space in constructing their *action-centric* graphs, where nodes represent actions instead of agents. They further assume anonymity and additivity of agent actions' effects, as found in congestion games (Rosenthal, 1973). The multiagent modeling framework introduced in the next chapter mainly exploits agent-centric structure, while making no such anonymity and additivity assumptions. I further seek to capture inter-agent dependencies induced from a wide variety of sources that are not necessarily restricted to payoff-related relationships of graphical games.

2.4 Probabilistic Graphical Models

Probabilistic graphical models, particularly undirected models such as MRFs (Kindermann and Shell, 1980), comprise the basis for the framework of graphical multiagent models that I develop and present in this thesis. First, denote an undirected graphical model as $G = (V, E)$, where V is a set of nodes corresponding to random variables in the set X, and E is a set of undirected edges. G represents a joint probability distribution compactly by exploiting the locality of relationships among variables. In particular, variables X_i and X_j are independent given the set of variables X' if there exists no path from node i to node j when all nodes corresponding to the variables in X' are removed from the set V (Koller and Friedman, 2009).

One can represent any given joint probability distribution with either a directed or an undirected probabilistic graphical model. However, there is no one-to-one transformation from a directed graphical model to an undirected graphical models that can preserve local probabilistic relationships (Murphy, 2001). Directed graphical models, such as Bayesian networks (Pearl, 2000), can capture scenarios where interactions have a natural directionality, typically from causal intuitions (Koller and Friedman, 2009). On the other hand, that undirected graphical models do not require one to ascribe directionality to the interaction between variables makes them a natural starting point for efforts to model multiagent interactions, which are often symmetrical.

Since the interactions between variables are undirected, one cannot assign for each node a conditional probability of the node given its neighbors, as in directed models. Instead, undirected models capture *affinities* between related variables through the concept of *factor*: a factor $\pi(X') \geq 0$ of a variable set X' denotes the affinity between variables in X'. Furthermore, a factor over a maximal clique[1] labeled c defines the *potential* of possible value assignments of variables in c. As a result, the joint probability distribution over all variables X can be factored into potentials of the set of all maximal cliques, denoted as $\mathcal{C}$, in the graphical model G:

$$\Pr(X) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \pi(X_c),$$

where Z is the normalization factor obtained by summing over all assignments of variables or nodes in X.[2] In general, factors over maximal cliques, referred as *potential functions*, do not have a local probabilistic interpretation, such as conditional or marginal probabilities. However, potential functions often have a natural interpretation such as “agreement," “constraint," or “energy," found in real-life domains, which motivates the application of undirected probabilistic graphical models in multiagent settings (Koller and Friedman, 2009).

In many scenarios, the modeler often faces the problem of learning dependence struc-

[1] A clique of a graph is a fully-connected subset of nodes. A maximal clique cannot be extended without violating the property of being fully connected.

[2] Section 3.1 discusses different approaches to approximately computing the normalization factor Z.

tures for probabilistic graphical models from observation data, as such structures are not available as input. In particular, one would like to search for a network structure that maximizes a score over the training data, such as data likelihood.[3] Due to computational complexity, most of the search methods in structure learning are local search algorithms, from greedy hill-climbing to simulated annealing. In addition to the problem of reaching local maxima, a local search also often encounters sets of network structures that have the same score, forming many plateaus in the search landscape (Koller and Friedman, 2009). To reduce the impact of local maxima and plateaus, one can choose to apply a variety of remedies, such as basin flooding, tabu search, beam search, and random restarts, depending on problem specifics and computational constraints. Heckerman et al. (1995) compared different search algorithms for the structure learning problem, and concluded that local search offers the best time-accuracy tradeoff. Some examined the combinatorial problem of searching over all network structures (Koivisto and Sood, 2004; Silander and Myllymaki, 2006), which opens the possibility for parallelization. A different line of research proposes local search over a different space of *I-equivalence* classes, which are subsets of networks that have equivalent score, and furthermore guarantees to learn the optimal (undirected or directed) Bayesian structure at the large data limit (Chickering, 2002; Nielsen et al., 2002).

In dynamic multiagent settings, the modeler is often interested in reasoning about the state of the world represented by the variables X as it evolves with time, denoted by t. As the space of all possible *trajectories*, assignments of values to variables X over time, is often very large and complex, representing a joint distribution over such trajectories entails making simplifying assumptions. *Dynamic Bayesian Networks* (DBNs) (Kanazawa and Dean, 1989), a well studied probabilistic model for dynamic systems, make two assumptions in particular: (i) variables at time t cannot depend directly on variables at time $t' < t - 1$ (Markov assumption), and (ii) the conditional probability of a variable X_i on a set of variables X' is the same for all t (stationary assumption) (Ghahramani, 1998). In

[3] As likelihood score does not have any properties that prevent overfitting, additional complexity constraints are needed to disallow overly complicated structure.

essence, a DBN is a probabilistic *directed* graphical model in which all copies of variable X_i^t for $t > 0$ have the same dependency structure, specifying the same conditional probability $\Pr(X_i \mid X_{CPD_i})$ for all t, where CPD_i includes variables in both time slices t and $t-1$. As a result, one can compactly represent the joint probability distribution over infinite trajectories by computing the time-invariant conditional probability distribution for each variable. Motivated by DBNs, the dynamic version of graphical multiagent models, introduced in Chapter IV, similarly adopts the stationary assumption, and the *semi*-Markov assumption where a variable in t may be dependent on variables in $t-1, \ldots, t-k$ for some fixed k. However, the presence of undirected edges in history-dependent graphical multiagent models violates the acyclic property of Bayesian networks, hindering the application of inference and learning methods in DBNs, such as Kalman smoothing (Kalman, 1960) and forward-backward algorithms (Smyth et al., 1997).

CHAPTER III

Graphical Multiagent Models

The need to accommodate different beliefs about agent reasoning other than game-theoretic reasoning (Chapter II) motivates the framework of *graphical multiagent models* (GMMs) (Duong et al., 2008). I first describe the details of GMMs, which are simply graphical models where the joint probability distribution is interpreted as an uncertain belief (e.g., a prediction) about agent actions. In particular, GMMs allow the expression of hybrid beliefs, constructed as a combination of multiple knowledge sources, as described in Section 3.4. I then use an example scenario of interdependence in technology adoption (Section 3.3) to illustrate the flexibility of GMMs by showing how to construct plausible multiagent models using two qualitatively distinct sources of belief about agent behavior. One example model is based on the game form of the scenario, and another based on heuristic assumptions of behavior. Computational experiments reveal the combined models to be superior to the individual knowledge sources in their ability to predict multiagent behavior generated by a reinforcement learning process.

3.1 Static GMMs

A (static) GMM is a graphical model, $\mathsf{G} = (V, E, A, \pi)$, with vertices $V = \{1, \ldots, n\}$ corresponding to the agents, each of which chooses an action a_i from its action domain A_i, and edges $(i, j) \in E$ indicating a local interaction between i and j. The graph defines

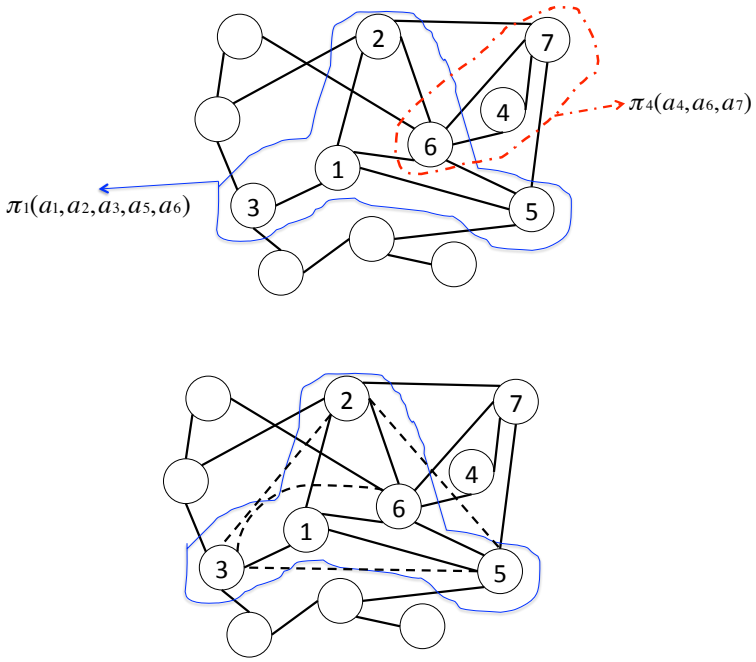

Figure 3.1: An example graphical multiagent model of 12 nodes connected by black solid straight-line edges. (Top) The potential associated with node 1 π_1 is defined over the actions of nodes in neighborhood N_1 (bounded by the light blue solid curve). The potential associated with node 4 π_4 is defined over the actions of nodes in neighborhood N_4 (bounded by the light red dotted curve). (Bottom) The triangulation of node 1's neighborhood entails adding the black dotted edges between its neighbors.

for each agent a *dependence neighborhood*, $N_i = \{j \mid (i,j) \in E\} \cup \{i\}$, including i and its neighbors $N_{-i} = N_i \setminus \{i\}$. Each dependence neighborhood is associated with a potential function $\pi_i(a_{N_i}; \theta_i) : \prod_{j \in N_i} A_j \rightarrow \mathbb{R}^+ \cup \{0\}$, where θ_i denotes the function's parameter vector. For simplicity in notation, I typically take θ_i as implicit and write $\pi_i(a_{N_i})$. Intuitively a local configuration of joint actions with a higher potential is more likely to be part of the global outcome than one with lower potential. One can then factor the joint probability distribution of agent actions into neighborhood potentials as follows:

$$\Pr(a) = \frac{\Pi_i \pi_i(a_{N_i})}{Z}, \tag{3.1}$$

where Z is a normalization term. An example GMM is depicted in Figure 3.1.

The derivation of the joint probability distribution in (3.1) as a normalized product of potentials follows a reduction procedure employed by Daskalakis and Papadimitriou (2006) in their mapping of graphical games to Markov random fields. Given the original graph $G = (V, E)$, one can add edges to construct the triangulated graph $G' = (V, E')$. As a result, each node i's neighborhood in the original graph is contained by a clique c_i in the triangulated G', by construction. Let $\mathcal{C} = \bigcup_i \{c_i\}$ be the set of cliques of G'. For instance, in Figure 3.1 (right), the neighborhood N_1, bounded by the light blue solid curve, would be transformed into a clique in the corresponding triangulated graph after the black dotted edges are added. Note that there might be $i \neq j$ with $c_i = c_j$. The potential of a clique c is defined as the product of neighborhood potentials $\pi_i(a_{c_i}) = \pi_i(a_{N_i})$ for all nodes i such that $c_i = c$. Thus, the Markov random field on the graphical model G' defines an unnormalized probability distribution p over joint actions, factored according to the cliques $\mathcal{C}$,

$$p(a) = \prod_{c \in \mathcal{C}} \prod_{i : c_i = c} \pi_i(a_{N_i}). \tag{3.2}$$

Since the potential of each node's neighborhood is included exactly once in (3.2), one can simply factor this unnormalized joint distribution into neighborhood potentials, resulting in the joint probability distribution in (3.1).

The size of the GMM description is exponential only in the size of local dependence neighborhoods rather than in the total number of agents. However, the complexity of computing the normalization factor Z in (3.1) is exponential in the number of agents n, and thus precludes exact inference and learning in large models. Regardless, that GMM is basically a probabilistic graphical model enables the exploitation of statistical inference tools from the full repertoire of graphical model algorithms.

There have been many studies on approximation algorithms for probabilistic graphical models for large network scenarios. One approach involves making use of Monte Carlo sampling methods (Neal, 1993; Dagum and Luby, 1993; Jensen et al., 1995), which are simple to implement but often slow to converge. Variational methods (Kappen and Ro-

dríguez, 1997; Xing et al., 2003; Weiss, 2001) are usually deterministic procedures that provide error bounds on the approximated probabilities. However, these methods are highly problem-specific, as there is no systematic approach on applying variational transformations on particular graphical models (Jordan et al., 1999).

I have adopted the *belief propagation* method for approximately computing Z (Broadway et al., 2000). Belief propagation is exact in tree-graph models, and has shown acceptably good results with reasonable runtime in sparse cyclic graphical models. This property makes belief propagation a viable approximation inference method for GMMs, especially for scenarios involving sparsely connected networks of many small-size neighborhoods. It is important to note that even approximation inferences on graphical models can still be prohibitively expensive in worst-case scenarios, and in fact are dependent on the graphical structure in terms of complexity.

Although the mayor election example in Chapter I describes a dynamic scenario, one can construct a static GMM that focuses on predicting the final election outcomes only, without any knowledge about how public opinions have changed during the days prior to the election. For example, the modeler may capture each voter's own party preference (declared party membership) and the utility gained from persuading others in the form of a payoff function, and subsequently incorporate the modeler's own belief about voter reasoning in computing the system's joint probability distribution of election outcomes. I refer to this static model of the mayor election scenario as the *simplified election model* henceforth.

3.2 Graphical Multiagent Model Examples

In this section I provide examples of GMMs reflecting strategic decisions in game settings, and agent interactions in scenarios without any implicit game structure.

3.2.1 GMMs for Computing Pure Strategy Nash Equilibria

One source of potential functions is solution concepts of a *graphical game* played by n agents. First, examine a graphical game, $\mathcal{G} = (U, \{A_i\}, \{J_i\}, \{u_i(.)\})$, as defined in Section 2.3. Daskalakis and Papadimitriou (2006) defined for graphical games a binary potential function, associating a high value for configurations where each agent's strategy choice is a best response to its neighbors (i.e., maximizes payoffs given a_{J_i}), and a low value $0 < \epsilon \ll 1$ for all other configurations:

$$\pi_i(a_i, a_{J_i}) = \begin{cases} 1 & \text{if } a_i \text{ is a best response to } a_{J_i} \\ \epsilon & \text{otherwise.} \end{cases} \tag{3.3}$$

Under this construction, the problem of finding a maximum a posterior estimation (MAP) configuration of the unnormalized joint probability (3.2) is equivalent to the question of whether the graphical game $\mathcal{G}$ has a pure strategy Nash equilibrium (Daskalakis and Papadimitriou, 2006):

$$\mathcal{G} \text{ has a PSNE} \Leftrightarrow (\max p(a) = 1).$$

In other words, if there exists some PSNE in $\mathcal{G}$, the resulting joint distribution function assigns near-zero probability to non-Nash action profiles, and thus specifies the set of PSNE. As a result, one can compute the set of pure strategy Nash equilibria in polynomial time for large classes of graphical games[1] by employing the *junction tree algorithm* (Lauritzen and Spiegelhalter, 1988) to the reduction's output, a Markov random field. Moreover, one can construct a GMM G that resembles the Markov random field MR for graphical game $\mathcal{G}$ in representational and computational capabilities by assigning $V^{\mathsf{G}} = I^{\mathcal{G}}$, $A_i^{\mathsf{G}} = A_i^{\mathcal{G}}$, $N_i^{\mathsf{G}} = J_i^{\mathcal{G}} \cup \{i\}$,[2] and $\pi_i^{\mathsf{G}}(a_{N_i}) = \pi_i^{\mathsf{MR}}(a_i, a_{J_i})$ for all i.

Consider the simplified mayor election model described in Section 3.1, where each

[1] Theses include previously known efficiently solved games, as well as graphical games with $O(\log n)$-tree width (Daskalakis and Papadimitriou, 2006).

[2] For simplicity, I retain this same assumption in other GMM examples.

town-person i votes for either the Democrat candidate ($a_i = 0$) or the Republican candidate ($a_i = 1$), and the main goal is to make predictions about outcomes without taking into account any intermediate public opinion polls. In this section I further impose that if their chosen candidate wins, voters get some positive utility, which, however, is overshadowed by the utility achieved by voting in sync with their family, friends, and community. For instance, this utility source can be defined for each voter i as: $u_i^{sync}(a_{N_i}) = \sum_{j \in N_i}[\zeta(1 - |a_j - a_i|)]^2$, which increases quadratically with the number of neighbors agreeing with i's final pick. Moreover, assume that voter k has been a loyal Republican: he receives utility of U_{high} when voting Republican, and U_{low} when having to settle for a Democrat candidate. His total utility for voting a_i, given others' actions $a_{N_{-k}}$, is computed as: $u_k(a_{N_k}) = u_k^{sync}(a_{N_k}) + U_{high}a_k + U_{low}(1 - a_k)$.

If May assumes that people of AA are perfectly rational and thus pick a Nash equilibrium outcome for the graphical game induced from their voting utility function, she can construct a GMM or a Markov random field as described. If a voter i chooses a suboptimal action $\hat{a}_i$ that is not the best response to his or her neighbors' actions $\hat{a}_{N_{-i}}$, the corresponding potential $\pi_i(\hat{a}_{N_i})$ for neighborhood N_i will assume near-zero value, which forces the joint probability for any outcomes that contain $\hat{a}_{N_i}$ to be negligible. As a result, if the number of NE, m_{NE}, is greater than 0, the joint probability of any MAP resulting from this model is then $\frac{1}{m_{NE}}$, as all non-Nash action profiles are assigned negligible values. Note that the aforementioned Markov random field mapping focuses only on identifying PSNE of a graphical game, but does not address the case where no PSNE exists. In the GMM framework, if there is no NE, the action profiles containing fewer agents that do not pick best-response actions have higher probability of occurrence. May can then compute the winning chance of a candidate by summing over probabilities of all the outcomes where the candidate wins a majority of votes.

3.2.2 Regret GMMs

A natural generalization of the Nash equilibrium potential function would smooth out the binary distinction, assigning intermediate potentials based on the *degree* to which agents deviate from their best responses. One may not wish to assume that agents play best responses with certainty, as they may not be perfectly rational, or the attributions of payoff functions may be inexact. Intuitively, as *regret* increases, agents are more apt to recognize and select better alternative actions. One may expect that high-regret joint actions are less likely to be played (all else equal), and capture this intuition in a *regret potential*,

$$\pi_i(a_{N_i}) = e^{-\epsilon_i(a_{N_i})\lambda_i}, \tag{3.4}$$

where λ_i, the *temperature* parameter, provides a way to calibrate the association between regret and relative likelihood. When $\lambda_i = 0$, agent i uniformly randomly picks its action regardless of payoff consequences, as $\pi_i(a_{N_i}) = 1$ for all a_{N_i}. Greater values of λ_i accord lower probability to agents making less than perfectly rational decisions. As $\lambda_i \to \infty$, the lowest-regret joint actions, in which agent i plays best-response to others in N_i, will obtain larger potential than other joint actions, effectively reducing others' regret potential to zero. Consequently, the regret potential function (3.4) approaches the binary potential function introduced for computing pure strategy Nash equilibrium (3.3).

In the aforementioned mayor election example, given the choices of k's neighbors, his regret caused by choosing a Democrat over a Republican is computed as: $\epsilon_k(a_{N_{-k}}, a_k = 0) = u_k(a_{N_{-k}}, a_k = 0) - u_k(a_{N_{-k}}, a_k = 1)$. If person k is well-informed and has been actively involved in political discussions, one expects him to pick the best option with high likelihood, implying a high λ_k value. On the other hand, any signs of k's ignorance about the candidates' background and platforms, as well as his tendencies to switch votes too frequently may prompt the modeler to choose a low $\lambda_k \geq 0$.

3.2.3 GMMs for Computing Correlated Equilibria

Kakade et al. (2003) argued that Markov networks are a powerful language for representing a subclass of all correlated equilibria of a graphical game. In particular, for all graphs G and all joint distributions p over joint actions a, there exists a distribution q that is representable as a local Markov network with graph G such that p and q yield the same expected payoff vector. In other words, for each i, $\mathbf{E}_{a\sim p}[u_i(a)] = \mathbf{E}_{a\sim q}[u_i(a)]$, and p and q are *payoff equivalent*. That GMMs belong to the class of local Markov networks, typically defined with potential functions ranging over settings of local neighborhoods in the graphs, implies their ability to express any correlated equilibria of a graphical game up to payoff equivalence. This representational power enables efficient computation of different correlated equilibria in time polynomial in the size of the graphical game, which grows exponentially with its largest node degree (Kakade et al., 2003).

3.2.4 Information Diffusion GMMs

GMM potential functions can also express agent choice patterns not derived explicitly from utility functions. In particular, I employ GMMs to represent a process of information diffusion in social networks (Duong et al., 2011), which is described in greater detail in Chapter VI.

Define a_i^t, node i's state at time t, to be 1 if it is infected with information at time t. The function $c(a_{N_i}^{t-1})$ indicates the number of nodes in the dependency neighborhood N_i that are infected before t. I adopt a model based on the *cascade* probability (Goldenberg et al., 2001; Kempe et al., 2003):

$$\chi(a_i^t = 1 \mid a_{N_i}^{t-1}) = \alpha + \beta \frac{c(a_{N_i}^{t-1})}{|N_i|}. \tag{3.5}$$

where β and α represent the tendency of infection from interacting with one's infected neighbors, and from sources other than neighbors.

The GMM representation of information diffusion employs a functional form for potentials based on this cascade model formula. First, define $g(a_y = 1 \mid a_Y)$ as a probability measure of any uninfected node y in the set of nodes Y becoming infected given the current a_Y: $g(a_y = 1 \mid a_Y) = \alpha + \beta c(a_Y)/|Y|$. Next overload $g(a_Y) = g(a_y = 1 \mid a_Y)$, as g is identical for all $y \in Y$. Let c be the number of nodes in N_{-i} that become infected in time t. The potential function of neighborhood N_i is the product of three terms:

$$\pi(a^t_{N_i} \mid a^{t-1}_{N_i}) = \chi(a^t_i \mid a^{t-1}_{N_i}; \alpha, \beta) g(a^{t-1}_{N_{-i}}; \alpha_1, \beta_1)^{|c-\gamma|N_i||} \left(1 - g(a^{t-1}_{N_{-i}}; \alpha_{-1}, \beta_{-1})\right)^{|N_{-i}|-c}. \tag{3.6}$$

A full description of this construct can be found in Section 6.3.1.2. Here I focus on the second term in (3.6). If one assumed independence of agent states, the exponent in the second term would simply be c, as the joint probability of c nodes from Y becoming infected is simply $g(s_Y)^c$. The distance of this count c from a fraction of neighborhood size captures correlation among neighbor infections, resulting in the exponent $|c - \gamma|N_i||$ with $\gamma \in [0, \infty)$, where γ dictates the correlations between agent states, with $\gamma = 0$ corresponding to complete independence.

3.3 Example: Technology Upgrade Scenario

In this section, I illustrate the GMM framework and motivate the problem of combining knowledge sources through a static multiagent system example. In the *Internet industry partnership* domain,[3] companies must decide whether to retain $(a = 0)$ or upgrade $(a = 1)$ their current technology. The payoff for each strategy depends on the choices of other companies to which they are related through some kind of partnership—their neighbors in the interaction graph. For example, the benefits of upgrading may be larger when one's partners also upgrade, since keeping their technologies synchronized enhances compatibility.

[3]The example is inspired by the network model of Krebs (2002), cited by Kearns (2002).

My study focuses on a fragment of the partnership network consisting of 10 representative companies, as depicted in Figure 3.2. Each node represents a company, which is characterized by three parameters: (i) size class, z (lower numbers corresponds to bigger sizes); (ii) sector, t: either commerce, content, or infrastructure; and (iii) change coefficient, $\mathsf{ch} \in \{1, 2, 3\}$, representing the intrinsic adaptability of the company's technology. The study by Krebs (2002) provides sector information and a rough order of size; the change coefficient is assigned in an arbitrary manner. Let J_i denote the set of neighbors or partners of company i. I construct a payoff function, u, defining the value of retaining or upgrading technology, given the actions of a firm's neighboring companies and the parameters describing these companies. Although many of the details of my specification are somewhat contrived, the scenario serves the purpose of demonstrating capabilities of the GMM approach.

3.3.1 Description

Qualitatively, payoff is increased by agreement with neighbors or partners, where larger partners from the same sector are relatively more important. I first introduce an intermediate value w_{ij} for each connected pair of companies, reflecting the strength of i and j's partnership:

$$\mathsf{w}_{ij}(a_i, a_j) = (\mathsf{z}_i + \mathsf{z}_j)\left(1 + \frac{\mathsf{y}_{ij}}{2^{I(t_i,t_j)} + 4^{1-I(t_i,t_j)}}\right)^{I(a_i,a_j)}, \tag{3.7}$$

where $\mathsf{y}_{ij} \sim U[0, 1]$ is assigned randomly, and $I(x, y)$ is an indicator function representing agreement in strategy and technology sector: $I(x, y) = 1$ if $x = y$, and 0 otherwise. The intuition for (3.7) is that the mutual influence between two firms increases with their size, agreement in action, and sector commonality. Also define function $\phi(\mathsf{ch}_i, a_i)$ to compare i's action to its change parameter ch_i. Specifically, ϕ's value is positive if a_i is upgrade (retain) and ch_i is greater (smaller) than 0.5. The interpretation is that a value greater (smaller) than

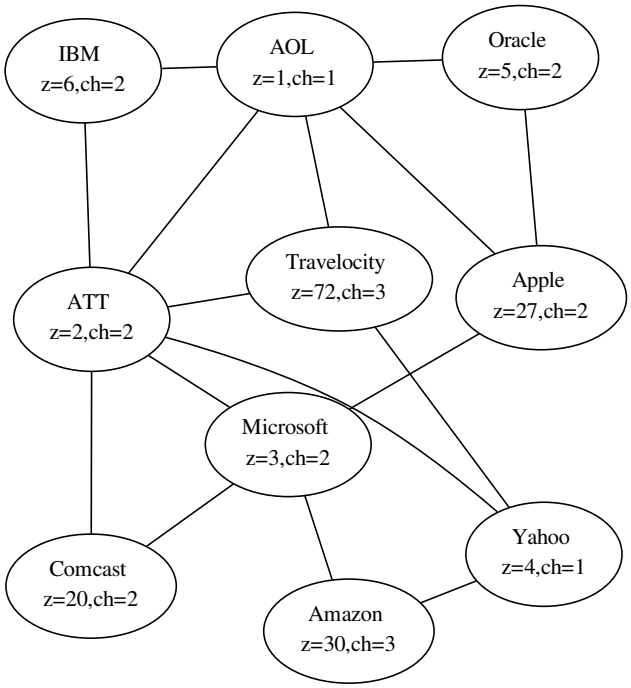

Figure 3.2: Part of the Internet industry partnership network, from Krebs (2002): $\mathsf{ch} \in \{1, 2, 3\}$ represents the intrinsic adaptivity of a company's technology, and z captures a company's size class.

0.5 for ch_i implies that i is flexible (inflexible) with respect to technology change:

$$\phi(\mathsf{ch}_i, a_i) = \begin{cases} \frac{a_i+1}{2} - \mathsf{ch}_i & \text{if } \frac{a_i+1}{2} - \mathsf{ch}_i < 0.5 \\ 0.5 - \frac{a_i+1}{2} + \mathsf{ch}_i & \text{otherwise} \end{cases}$$

Finally, the overall payoff combines the pairwise partnership weights, further adjusted by the company's flexibility in upgrading its technology:

$$u_i(a_i, a_{J_i}) = (1 + \mathsf{y}_i \phi(\mathsf{ch}_i, a_i)) \sum_{j \in J_i} \mathsf{w}_{ij}(a_i, a_j),$$

with $\mathsf{y}_i \sim U[0, 1]$.

3.3.2 GMM Constructions

The graphical structure on which the following GMMs are based is presumably the same as the input partnership graph. In other words, agent i's GMM neighborhood N_i is identical to the union of its partnership set J_i and $\{i\}$ for all i. Given the payoff function u, one can generate a regret potential function (3.4) in a straightforward manner, parametrized by temperature. This potential function in turn defines a regret GMM, reG.

Also define a second GMM, rbG, corresponding to a simple rule-based agent model without direct reference to the payoff function. In this model, each company independently applies a local heuristic to stochastically choose its action. Specifically, agent i changes its technology with probability $pChange(i)$, where

$$pChange(i) = 0.5(1 - 10^{-3})^{|N_i|}(1 - 10^{-3}\mathsf{z}_i).$$

Here let z be the size of company i. The intuition behind this heuristic is that the more partners ($|N_i| - 1$) a company has and the greater its size z_i, the less likely it is to change. Given the $pChange$ values, it is straightforward to define a potential function π_i for rbG, such that the outcome distribution is the same as generated by applying the rule independently for each company. In light of this independence, π_i is a function of only a_i, rather than a_{N_i}.

The two GMMs, reG and rbG, are based on qualitatively different sources of knowledge. If one believes that agents are essentially rational and aware of their environment, one may expect the regret GMM reG to predict behavior well. If instead one has evidence that agents choose heuristically based on partnership density and size, one might have greater confidence in predictions based on rbG, or on some other heuristic models that capture some intuition about agents' behavior. In other situations, one may consider that both models capture factors determining behavior, and view the knowledge sources as complementary.

3.3.3 Reinforcement Learning Simulation

The role of the reinforcement learning simulation model is to generate action data from a plausible agent interaction process. I treat the generated data as the actual outcome, and use it to evaluate the reG and rbG GMMs as well as combined models, introduced in Section 3.4. The simulation is based on the idea that actual agent behavior is generated by agents who start to play heuristically, then update their behavior over repeated interactions through a reinforcement learning (RL) process. Thus, the main task is essentially to predict the outcome of an RL regime: computing a reasonable estimate of the joint distribution of the agents' actions.

In the model, each agent is an independent learner, employing an RL procedure designed for partially observable environments (Jaakkola et al., 1995). The environment for company i comprises i and its partners (i.e., its neighbors J_i), and each configuration of partner strategies a_{J_i} is a possible state. The agent seeks to learn a stochastic policy: $\sigma_i(a_i \mid a_{J_i})$, which denotes the probability of choosing action a_i at state a_{J_i}. However, the agent does not actually observe a_{J_i} before taking its action. Thus, the action is actually selected based on the policy, for $\mathbf{a} \in \{0, 1\}$,

$$\Pr_i(a_i = \mathbf{a}) = \sum_{a_{J_i}} \sigma_i(a_i = \mathbf{a} \mid a_{J_i}) \Pr(a_{J_i}). \tag{3.8}$$

$\Pr(a_{J_i})$ is a stationary probability distribution over states, which, for simplicity, I initially set to be uniform.

To learn the policy σ_i, I apply the RL procedure by Jaakkola et al. (1995).

1. Initialize σ_i to $pChange(i)$ (i.e., the heuristic policy from rbG).

2. Generate action a_i using (3.8) for all i. Observe the resulting local state a_{J_i} and receive as reward the payoff $u_i(a_i, a_{J_i})$. Update $Q_i(a_i, a_{J_i})$, the average reward for taking action a_i in the associated local state.

3. Choose $\sigma_i^*(a_i \mid a_{J_i})$ to maximize $Q_i(a_i, a_{J_i})$. Adjust the current policy σ_i in the direction of σ_i^*: $\sigma_i \leftarrow \sigma_i(1-\gamma) + \sigma_i^*\gamma$, where γ is the learning rate. Update $\Pr(a_{J_i})$.

4. Repeat steps 2 and 3 until convergence.

For these experiments, I used $\gamma = 0.2$ and iterated steps 2 and 3 above 40 times, the point after which few changes occurred in the learned RL policy σ. Denote the model corresponding to RL simulation by simM.

Note that the RL process starts with the local heuristic rule-based policy, but is updated based on payoff experience. Thus one may expect that both the heuristic rule-based model and the rationalistic regret-based model may offer value for predicting the outcomes of simM. Note that although knowledge about both the heuristic starting point and game-theoretic equilibria is relevant, neither directly captures nor corresponds to the RL outcomes.

3.4 Knowledge Combination

Given two complementary sources of knowledge, I demonstrate how to integrate them into a single GMM. I formulate the problem for the case that the two knowledge sources are expressed explicitly as two different GMMs, G_1 and G_2. As some of the combination methods are naturally described in terms of taking data as input, I start by describing how to generate such a data set from a GMM, and likewise how to induce GMM potential function parameters from data. This duality between the two representations enables one to apply the combination methods in general settings where knowledge sources come in either form.

3.4.1 Learning and Sampling

One can measure predictive performance of a GMM G with respect to a data set $D = \{a^1, \ldots, a^m\}$ of m joint actions according to the log-likelihood of the data under the model:

$$\text{Score}(\mathsf{G} \mid D) = \sum_{k=1}^{|D|} \log \Pr_{\mathsf{G}}(a^k).$$

Consider the problem of learning GMM parameters θ from data D in order to maximize this score (Kappen and Rodríguez, 1997): $L(D \mid \theta) = \text{Score}(\mathsf{G} \mid D)$, assuming that the graphical structure is given as input. I employ the gradient ascent method with random restarts (as well as early stopping for more complex problems in later chapters) to maximize data likelihood, which entails computing the gradient of L w.r.t θ:

$$\begin{aligned} \nabla\theta &= \frac{\partial L(D|\theta)}{\partial \theta} \\ &= \frac{\sum_{k=1}^{m} \sum_i \partial \log \pi_i(a^k_{N_i})}{\partial \theta} - m \frac{\partial \log Z}{\partial \theta}, \end{aligned} \tag{3.9}$$

and adjusting $\theta \leftarrow \theta + \alpha \nabla\theta$, where α is the learning rate, until the gradient is below some threshold.

A major problem in graphical-model parameter learning is the intractability of calculating $\log Z$ in (3.9). It entails iterating over all possible outcomes of the game, which is exponential in the number of players n, rendering exact inference and learning in undirected graphical models intractable. Since this study's priority is to accurately evaluate knowledge combination methods for GMMs, my implementation of the inference and learning algorithms does not employ any approximation. The incorporation of generalized belief propagation (Yedidia et al., 2001) for approximating the normalization factor in GMM will be implemented and reported in later chapters.

Given a GMM G with known potential function parameters and graphical structure, one can also generate a data set of m joint actions from G. Since G specifies a joint probability distribution over joint actions for the n agents, I independently sample m joint actions a

directly from G to construct $D = \{a^1, \ldots, a^m\}$.

3.4.2 Combination Methods

I next outline three methods for combining two sources of knowledge, represented as G_1 and G_2, into a single GMM. Although these methods take GMMs as input, they also have the ability to translate input GMMs into a different form, such as action data, that most suits them.

Direct Update

The *direct update* method combines the two sources of knowledge, G_1 and G_2, into a new GMM, $\mathsf{directG}$, derived by tuning the θ_{G_1} parameters of G_1 to maximize predictive performance with respect to G_2. In particular, one may generate a data set from G_2 (as detailed in Section 3.4.1), and tune G_1 to maximize the likelihood of the generated data.

Opinion Pool

The *opinion pool* method constructs a combined model OPG, which is an aggregation of G_1 and G_2 into a single probability distribution:

$$\Pr_{\mathsf{OPG}}(a) = f(\Pr_{\mathsf{G}_1}(a), \Pr_{\mathsf{G}_2}(a)).$$

Note that the above equation does not entail defining for each agent a pooled potential function, but specifies a pooled probability function from the two inputs' joint probability distribution functions. The rationale is that potentials are not normalized like the joint probability, and thus, their absolute values contain little meaning when taken out of the context of their corresponding models. I adopt for the aggregation function the weighted

geometric mean, called the *logarithmic opinion pool* (logOP).

$$\Pr_{\mathsf{OPG}}(a) = \frac{\Pr_{\mathsf{G}_1}(a)^w \Pr_{\mathsf{G}_2}(a)^{1-w}}{Z}.$$

The logOP is the only pool known to preserve independence structure in graphical models (Pennock and Wellman, 2005), which is an important property in this context. The weight parameter w can be set by fiat, or tuned with some data set $\bar{D}$ to maximize the objective function:

$$L(\bar{D} \mid w) = \sum_{k}^{|\bar{D}|} \log \Pr_{\mathsf{OPG}}(a^k). \tag{3.10}$$

In brief, given the two components G_1 and G_2, the opinion pool method first initializes $w = 0.5$. It then repeats computing the gradient ∇w of the log-likelihood with respect to w by differentiating (3.10), and updating $w = w + \beta \nabla w$, where β is the learning rate, until the gradient is acceptably small.

Mixing Data

The *mixing data* method samples joint actions from the given GMMs G_1 and G_2 to generate data sets D_1 and D_2, respectively. It combines D_1 and D_2 into one data set mD by sampling from the two sources equally, though one could easily weight one source more than another by adjusting the sampling ratio. Subsequently, a new GMM mixG for a given parametrized form is induced by finding the parameters θ that maximize the likelihood of the new data mD.

Below is the outline of the mixing data method, which produces the combined GMM mixG.

1. Generate a sample of joint action outcomes D_1 and D_2 from G_1 and G_2, respectively.

2. Build the mixed data set mD from D_1 and D_2 with sampling ratio $\omega = 0.5$.

3. Initialize mixG with some λ_{mixG}. Update λ_{mixG} as in direct update using the data

mD.

4. (optional) Tune ω in the direction determined by the gradient-descent method to maximize mixG's performance on a small held-out part of the testing data. Repeat steps 3 and 4 until the gradient is below some threshold.

3.4.3 Experiments

I evaluate the above combination approaches experimentally using a simplified version of the Internet industry partnership domain. I concentrate mainly on Example 1, which is the scenario depicted in Figure 3.2. In the first experiment, I also examine Example 2, which employs a smaller graph including only the top five performing companies.

These experiments seek to combine the two previously defined GMMs: reG and rbG, and evaluate my combination approaches based on their ability to predict a test data set D^* derived from the RL-based model simM. In my implementation, rbG is represented by a data set D of joint actions generated by the heuristic rule-based model. I compare the performance of a model that combines knowledge sources combinedG relative to the performance of a baseline model baseG[4] using a ratio of scores, $R = \frac{\text{Score}(\text{baseG}|D^*)}{\text{Score}(\text{combinedG}|D^*)}$. The inverted order of baseG and combinedG is due to Score's negativity, and thus, any $R > 1$ indicates combinedG's improvement over baseG.

I experiment with several environment settings. For each setting, I conduct 20 trials, each of which involves a training data D set of 500 joint actions from rbG and a testing data set D^* of 500 joint actions from simM. Thus, each data point presented here is averaged over 20 ratio scores R.

3.4.4 Results and Analysis

Figure 3.3 displays predictive performance across the two examples (1 and 2) and the two baseline models (reG and rbG). Mixing data is consistently best of the three com-

[4]In the result section, baseG can be either rbG, reG, or simM, depending on the experiments' details.

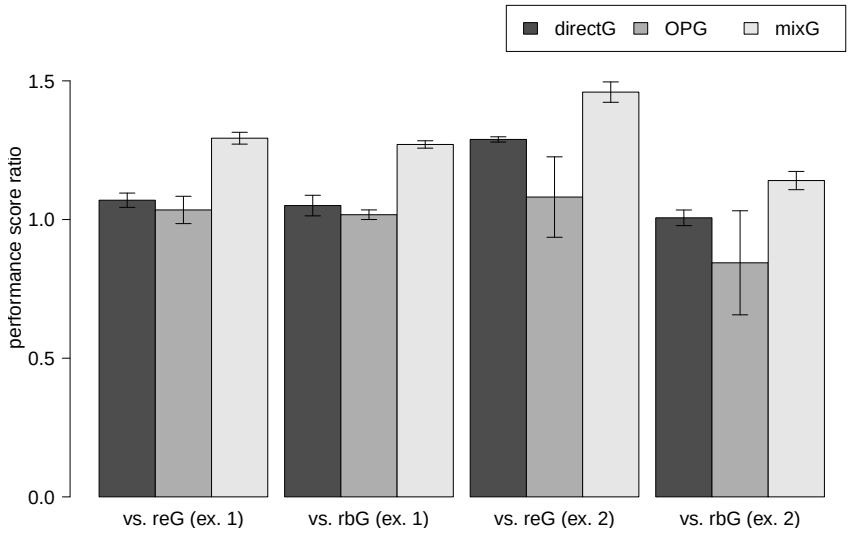

Figure 3.3: Three combined models directG (direct update), OPG (opinion pool), and mixG (mixing data), generate generally better predictions than either input model (baseline) in the two example scenarios.

bination methods, and direct update performs relatively better than opinion pool. All three methods yield better results than individual input models, suggesting that combining the two knowledge sources is beneficial regardless of which of the proposed methods is adopted.

Figure 3.4 shows the performance of various models compared to a model derived from the same gold-standard source simM as my test data D^*. A separate data set D' is sampled from simM and subsequently employed in learning a GMM simG of the parametrized regret form (3.4), using maximum likelihood to adjust the temperature parameter. The results reveal that the combined models, especially mixG, closely match simG in terms of predictive performance.

Next, I study the effect of varying the quality of the two input sources. Figure 3.5 shows the effect of varying the amount of joint-action data available in D. Specifically, I make a fraction $\rho|D|$ of observations, $\rho \in [0, 1]$, available to the three combination methods. From Figure 3.5 shows as long as $\rho > 0.1$, performance remains fairly stable. In these experiments, this corresponds to a threshold data set size of approximately $500 \times 0.1 = 50$. When the amount of data goes below this threshold, the combined model may be inferior

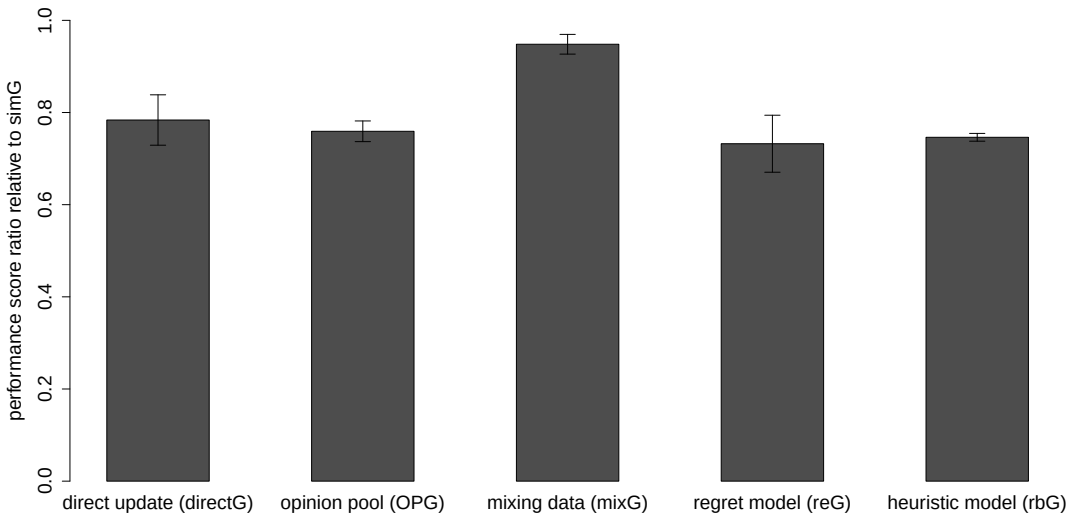

Figure 3.4: Prediction performance of the three combined and two input models versus that of the underlying model.

to the models reG and rbG.

Figure 3.6 shows the effect of varying the quality of the regret GMM provided as input to the combination methods. To modulate the accuracy of reG, I introduce a parameter δ controlling the relation of its temperature parameters to those of simG. Specifically, I set λ_{reG} to $(1+\delta)\lambda_{\text{simG}}$. When compared with the unchanged heuristic model rbG, the combination models show a slight decrease in their relative performance with δ, which reflects the effect of reG's inaccuracy on the combination methods. When the baseline is reG, in contrast, the degradation of the combined model is dominated by the effect of compromising the baseline reG. In other words, combining knowledge sources effectively compensates for degrading one of them.

In these experiments, OPG's poor performance relative to that of directG and mixG may be due in part to its reliance on only a single parameter, w, compared to the vector λ available in the other methods. Unlike directG that only employs reG at its initialization stage, mixG directly samples from reG and thus is able to employ information contained in reG more effectively. This differentiation likely results in the overall superiority in predictive performance of mixG over directG.

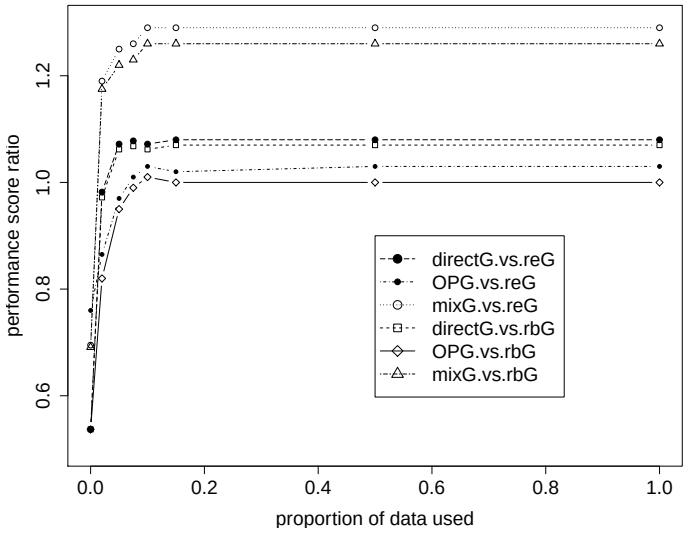

Figure 3.5: The combined models deteriorate sharply if the amount of training data falls below some threshold, while remaining relatively stable above that same threshold.

Figure 3.7 presents the results of an experiment designed to strengthen the claims about the benefits of integrating knowledge sources in a single model. I examine the combined models' performance in environments where the simulation model simM is not the product of RL that starts with the input model rbG. In particular, define a different heuristic model rbG′, such that $pChange_{\mathsf{rbG}'}(i) = 0.05$ for all i. Let E be the input data set generated from rbG′. Based on rbG′, I subsequently build the simulation model simM′ and its corresponding test data E^*. Given these data sets and models, one can evaluate the combined models across drastically different starting points, by comparing their performance when different input sources, D and E, are provided, on the same testing data (either D^* or E^*). First, I compare the performance of the heuristic models employed in generating input data, $\mathsf{rbG}_{(D)}$ (induced from rbG) and $\mathsf{rbG}_{(E)}$ (induced from rbG′): $\mathsf{rbG}_{(D)}$ performs 60% better than $\mathsf{rbG}_{(E)}$ when tested on D^*, whereas $\mathsf{rbG}_{(E)}$ outperforms $\mathsf{rbG}_{(D)}$ by 54% on E^*. This assessment affirms that the two different input data sets, D and E, are indeed differentiable in terms of the behavior models they represent.

The results presented in Figure 3.7 indicate that better performance is achieved in cases where the test and input environments coincide (the first and third measures) compared to

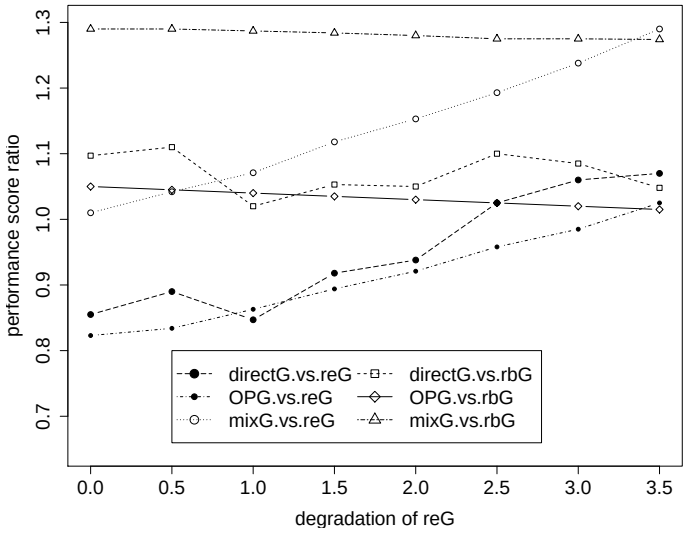

Figure 3.6: Combination methods compensate for the degradation of the input regret model, as controlled by δ: their performance exhibits robust gains against the degraded model reG, while showing only a slight decrease in comparison with the other input rbG.

those in which they are unrelated (second and fourth). This observation confirms that combined models whose input data contains some information about the underlying behavior model outperform those extracting irrelevant information from its noisy inputs. The gaps in mixG's performance between scenarios where test cases are derived from the same input (first and third) and from an unrelated model (second and forth) are relatively smaller than those in the other methods. This phenomenon is likely a result of mixG's more effective usage of both knowledge sources, which limits the impact of irrelevant input data, and possibly contributes to the good performance of $\mathsf{mixG}_{(D)}$, which is tuned to fit D, when tested on E^*.

In summary, the basic finding in this study is that combining two knowledge sources in this scenario does improve predictive power over either input source alone. Note that no details or properties of this technology upgrade scenario preclude the generalization of my empirical findings in other scenarios. I have also identified the most effective combination method among those tried—mixing data—and the existence of a threshold for data availability that can help boost efficiency. Furthermore, I have found that these three knowledge

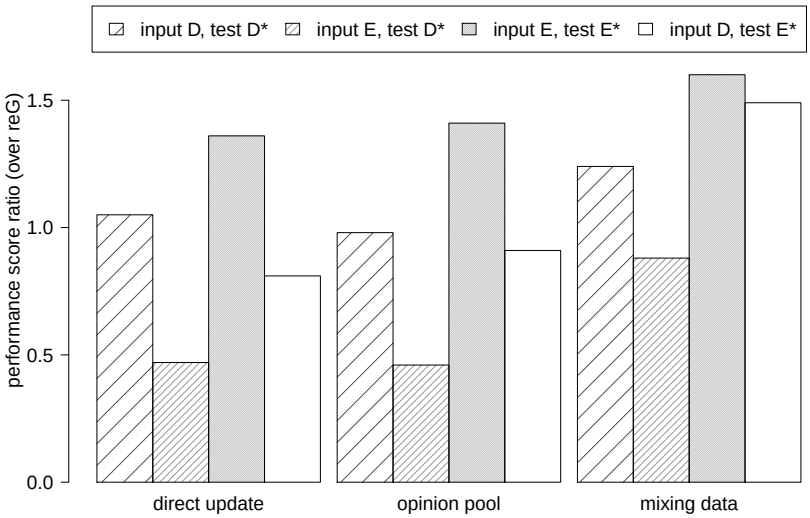

Figure 3.7: Combination methods generally produce better-than-baseline prediction results in settings where input models truly reflect part of the underlying model (input E with test E' or input D with test D'). As expected, these combined models fail to improve on input models when input does not capture underlying behavior.

combination approaches, especially mixing data, can effectively match the performance of modeling the reinforcement learning process directly.

CHAPTER IV

History-Dependent Graphical Multiagent Models

The prevalence of dynamic multiagent scenarios in real-world problems has motivated many models for dynamics in multiagent systems. Although a static GMM can answer certain queries about a dynamic system's outcomes without taking into account transient behaviors, such a modeling effort would have to rely on prior knowledge and assumptions about the system, which can be highly subjective and error-prone. In other words, leaving out information on transient behaviors, which are often manifested through time-series observational data, can result in arbitrarily inaccurate prediction outcomes. Moreover, beside final outcomes, the system modeler may be interested in making inferences about intermediate system states, explicitly mandating the inclusion of data on dynamic agent interactions and actions.

This study addresses the problem of directly capturing dynamic behavior over time: incorporating information on transient behaviors in answering inference queries about both the system's final outcomes and its intermediate states. My approach is to extend the static GMM framework to condition on history, creating *history-dependent graphical multiagent models* (hGMMs) (Duong et al., 2010, 2012). Like static GMMs, history-dependent GMMs allow direct specification of joint behaviors. Thus they can capture correlations in agent actions, without full specification of the system history of agent interactions. This treatment highlights a basic distinction between representations that specify *individual* multiagent

behavior, and those (such as hGMMs) directly expressing *joint* multiagent behavior. I illustrate the distinction with an example dynamic multiagent scenario and discuss the potential advantages of hGMMs over existing individual behavior modeling methods. I then empirically demonstrate the representational capabilities of hGMMs by comparing their prediction performance with that of individual multiagent behavior models in the example scenario of consensus dynamics, introduced by Kearns et al. (2009).

4.1 Formalism

This objective of this study is to model how agents interact in dynamic settings, where agents can change their actions over a time period of length T, as described in Section 2.1. Note that the only dynamic elements modeled here are agent actions. A *history-dependent graphical multiagent model* (hGMM) (Duong et al., 2010, 2012), hG, retains all the elements defined in the static GMM framework: V, E, A, and π, similarly defining a dependence neighborhood for each agent i: $N_i = \{j \mid (i,j) \in E\} \cup \{i\}$. In addition, the hGMM representation incorporates dynamics by conditioning joint agent behavior on an abstracted history of actions H^t. Formally, I introduce a *conditioning set* Γ_i for each agent i, denoting the set of agents whose prior actions condition this agent's potential function: $\pi_i(a^t_{N_i} \mid H^t_{\Gamma_i})$, where $H^t_{\Gamma_i}$ captures the subset of history relevant to i's probabilistic choice of next action.[1] Note that the dependence neighborhood N_i governs only the within-time probabilistic dependencies of node i, while the history available to agent i, $H^t_{\Gamma_i}$, is the subset of H^t pertaining to agents in Γ_i. One may expect N_i and Γ_i to overlap, although they capture two different types of dependence.

The central assumption here is that both *within-time edges*, which define dependence neighborhoods, and *across-time edges*, which specify the conditioning of current actions on history, are fixed across all time periods. Learning time-dependent graphical structures would be enormously more complicated and computationally expensive. However, the

[1] All these dependencies are from the perspective of the system modeler.

mechanics of inference tasks would basically stay unchanged even in scenarios where dependence neighborhood N_i^t (conditioning set Γ_i^t) at time t may be different from $N_i^{t'}$ ($\Gamma_i^{t'}$) at time $t' \neq t$.

An example hGMM structure is depicted in Figure 4.1. In this example, the abstraction function, denoted as $f(a_{\Gamma_i}^{t-1}, a_{\Gamma_i}^{t-2})$, can be designed to capture various statistics of the two most recent time periods, such as the frequency of a specific action among all agents in Γ_i in the past, the number of times i executed the same action with the majority of Γ_i, or simply the most frequently chosen action by all agents in Γ_i. Each agent i's choice of next action is conditioned on some abstraction of past actions by agents in the conditioning set Γ_i, which entails $H_{\Gamma_i}^t = f(a_{\Gamma_i}^{t-1}, a_{\Gamma_i}^{t-2})$.

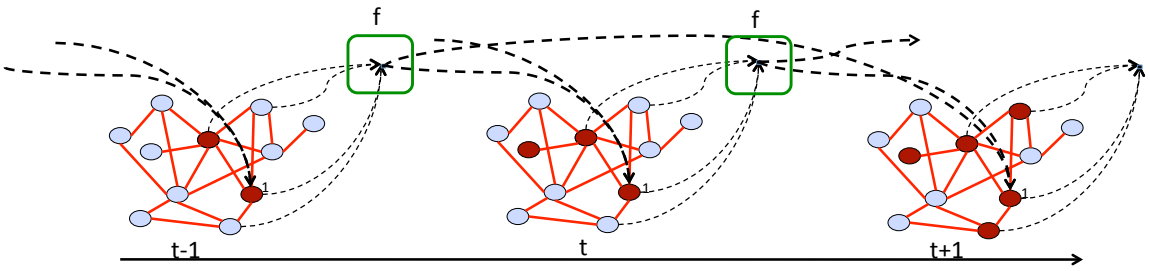

Figure 4.1: An example hGMM over three time periods of a dynamic scenario where agents choose between two different actions labeled red (light shaded) and blue (dark shaded). Undirected edges capture correlation among agents at a point in time. Directed edges (shown here only for agent 1) denote conditioning of an agent's action on some abstraction, encoded in function f, of others' past actions.

Each agent i is associated with a potential function $\pi_i(a_{N_i}^t \mid H_{\Gamma_i}^t)$: $\prod_{j \in N_i} A_j \to \mathbb{R}^+ \cup \{0\}$. The potential of a local action configuration specifies its likelihood of being included in the global outcome, conditional on history. As for static GMMs (3.1), the joint action distribution at time t is the product of neighborhood potentials:

$$\Pr(a^t \mid H^t) = \frac{\prod_i \pi_i(a_{N_i}^t \mid H_{\Gamma_i}^t)}{Z^t}. \tag{4.1}$$

As Z^t grows exponentially with the number of agents n,[2] my implementation of hGMMs employs the package libDAI (Mooij, 2008) for approximating Z^t using the belief propagation algorithm (Broadway et al., 2000). To simplify exposition, I drop the superscript t in all normalization factors henceforth. The joint distribution of a *trajectory* of agent actions over a certain time interval T is then computed as:

$$\Pr(a) = \prod_{t \in [0,\ldots,T]} \Pr(a^t \mid H^t).$$

Unlike Dynamic Bayesian networks (Kanazawa and Dean, 1989; Ghahramani, 1998), history-dependent graphical multiagent models in their current form support only forward inferences: employing history up to time t in computing joint, conditional, and marginal probabilities of agent actions executed after t, as discussed in Section 2.4. Adding latent (hidden) variables to the current form of hGMMs would provide the system modeler with greater modeling flexibility, but require new algorithms for learning the model and computing both forward and backward inferences.

Although this description of hGMMs focuses exclusively on dependence relationships between agent actions, the hGMM construct can incorporate other dependence representations. For instance, one may specify the dependence of person i's current candidate choice on the prominent town newspaper's endorsement of that candidate in the previous week. One can in principle represent such environmental factors as nodes in hGMMs, and capture their influence by across-time edges originating from these nodes to agent action nodes. These inclusions does not always entail adding within-time edges to the dependence graph structure of hGMMs, thus increasing the model's complexity by negligible amount.

[2]One needs to compute a different normalization factor Z^t specific to each time period t.

Learning Parameters

I address the problem of learning the parameters of an hGMM hG given the underlying graphical structure and data in the form of a set of joint actions for m time steps, $X = (a^0, \ldots, a^m)$. For ease of exposition, let θ denote the set of all the parameters that define the hGMM's potential functions. I use gradient ascent to search for the setting θ that maximizes the log likelihood of X:

$$L_{\mathsf{hG}}(X;\theta) = \sum_{k=0}^{m-h} \ln(\Pr{}_{\mathsf{hG}}(a^{k+h} \mid (a^k, \ldots, a^{k+h-1})); \theta)). \tag{4.2}$$

Note that as hGMMs in the current form do not support inferences with hidden variables, there is no need for an expectation maximization approach to learning model parameters, as in some dynamic Bayesian networks.

4.2 Individual Behavior Models versus Joint Behavior Models

Models of agent behavior on social networks provide some of the clearest examples for representing multiagent scenarios as stochastic dynamic systems. For example, Granovetter (1978) specifies an agent's strategy as a threshold function over the choices of other agents. This has been generalized and extended by Kleinberg (2007) to model the cumulative effect over time of an agent's neighbors' actions on its own behavior. For the consensus dynamics experiments (Kearns and Wortman, 2008), a simple multiplicative model that combines vote preferences with trends in neighbor actions can provide an effective probabilistic model of subject behavior (Kearns et al., 2009). In a study of related coordination games, Shoham and Tennenholtz (1997) proposed a rule whereby agents choose the action that maximizes their cumulative reward over a recent interval, and characterized the convergence to conventional behavior. Information about other agents' historic actions has also been shown instrumental in predicting learned behavior in a simple repeated game (Mookherjee and Sopher, 1994).

Though varied in complexity and approach, the above models share one common feature: the behavior of the multiagent system is the product of separately specified individual behaviors. That is, for each agent one can specify a probabilistic strategy $\sigma_i(H^t) = \Pr(a_i^t \mid H^t)$. In such a formulation, agent interactions are captured by the conditioning of individual behavior on complete history. The agents' actions are probabilistically dependent, but conditionally independent given this common history, yielding the joint distribution

$$\Pr(a^t \mid H^t) = \prod_i \sigma_i(H^t). \tag{4.3}$$

I refer to a dynamic multiagent model expressible by (4.3) as an *individual behavior multiagent model*. Individual behavior multiagent models provide an intuitive form for expressing collective behavior as a function of individuals. Their key premise is that the common observed history is sufficient to capture correlations in agent behavior. Although this may be a valid assumption in principle, it is usually infeasible to condition on the entire history in a practical specification of agent behavior.

Without the assumption of conditional independence given history, one generally needs to specify the *joint behavior* $\Pr(a^t \mid H^t)$ of multiple agents directly. This approach quickly becomes intractable, as the size of such a specification grows exponentially in the number of agents. However, if the interactions among agents can be localized (given history), one may be able to exploit this structure to achieve a more compact representation of the joint distribution. The history-dependent graphical multiagent model specified by (4.1) is an example joint behavior model. For each i, the potential function $\pi_i(a_{N_i}^t \mid H_{\Gamma_i}^t)$ directly models the joint behavior of the dependence neighborhood N_i, making no assumption of conditional independence given history. At the same time, hGMMs can take advantage of the locality of agent interactions, as encoded in the graphical structure that defines hGMM dependence neighborhoods.

One can interpret $H_{\Gamma_i}^t$ generally as a summary or abstraction of local history with re-

spect to N_i, since finite memory and computational power often preclude complete retention of historic observations. Moreover, from the perspective of the system modeler, only a partial view of the full history may be available. Yet another motivation for abstraction is provided by the need to limit complexity in order to effectively learn the model from a limited amount of data. Given an abstracted history representation, agent decisions generally appear correlated, even if they are independently generated conditional on the full history.

The emergence of correlated behavior is illustrated as a result of history abstraction through a two-agent example of the example dynamic setting depicted in Figure 4.2. Agents 1 and 2 choose actions independently given complete history. For instance, at time t, agent 1 chooses a_1^t based on its observations of both agents' past actions at time periods $t-1$ and $t-2$, but independently from agent 2's choice a_2^t. It follows that with respect to any substantive abstraction (e.g., abridgment or summarization) of the conditioning set $\{a_1^{t-2}, a_2^{t-2}, a_1^{t-1}, a_2^{t-1}\}$, a_1^t and a_2^t are no longer conditionally independent. In other words, abstraction of history induces correlations among agent actions, hence the imperative for joint behavior modeling.

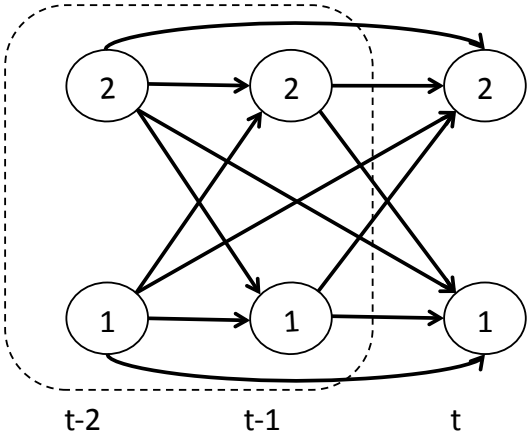

Figure 4.2: Agent 1 and 2 choose actions independently, conditional on full history (enclosed by the dotted line boundary). In other words, there exists no path between node a_t^1 and a_t^2 that does not pass through any node outside of the dotted-line enclosure.

4.3 Example: Consensus Dynamics Scenario

I illustrate the problem of representing dynamic multiagent behavior with a problem closely resembling the mayor election example outlined in Chapter I, where voters attempt to persuade others in their social circles to pick the same mayor candidate. In particular, the focus in this study is a *consensus dynamics* scenario which was introduced and studied by Kearns et al. (2009), and can be modeled as a game played on a network. Similar to the mayor election example, the consensus dynamics scenario allows each agent to choose between two available actions (vote options), labeled red (0) and blue (1). The scenario terminates when all agents' votes agree, or at the time limit T if no consensus is reached by then. In the asynchronous version of the scenario, agents may change their votes at any time, until the termination condition is reached. This study also examines a discrete-time version, where opportunities to update votes occur at a finite number of points. Upon termination, each agent i receives an individual reward $r_i(a) > 0$ if everyone converges on action a, and nothing otherwise. The variation in reward functions by agent reflects differing relative preferences for the available options. Despite these preference differences, agents have a common interest in achieving consensus, as without a unanimous vote nobody gets a reward.

Another pivotal feature of the consensus dynamics scenario is that agents have limited knowledge of the others' current votes. Specifically, agents are connected in an *observation* graphical structure, and can observe the votes of only their neighbors in the graph. In addition, agent i knows only its local observation graph structure, namely its neighbors N_i^{O}, the degree of each neighbor $k \in N_i^{\mathsf{O}}$, and edges between its neighbors.

Kearns et al. (2009) conducted several human-subject experiments based on the consensus dynamics scenario, each of which is assigned with an observation graph structure and a payoff function. The researchers were particularly interested in examining people's ability to make collective decisions with limited communication, as well as the role of the observation graph structure in determining final consensus outcomes.

Figure 4.3 shows what an agent observes during a typical experiment. The game here is displayed from the perspective of agent I, whose payoffs are $\$0.75$ and $\$1.25$ for 0 and 1 consensus outcomes, respectively. Agent I can observe only the agents to which it is connected in the graph—namely II, III, and IV, but not V and VI, shown in the dotted portion of the graph. In particular, the agent knows each neighbor's node degree (the number shown in parenthesis), the connections among its neighbors, and their current votes of either 0 or 1. If one assumes the length of each discrete time period is one second, and agent memory has a very short time horizon, $h = 1$, then at time period $t = 10$ agent I's history H_I^{10} consists of $\{(a_I^9 = 1, a_{II}^9 = 0, a_{III}^9 = 1, a_{IV}^9 = 1)\}$. One possible action plan for agent I is to vote in agreement with the majority of its neighbors from the last time period, in which case $a_I^{10} = majority(H_I^{10}) = 1$.

Figure 4.4 further illustrates the dynamic behavior in an example experiment instance, where a small group of well-connected red-favoring agents are centrally positioned in a network of mostly loosely connected blue-favoring agents. Though most of the nodes in the network initially chose blue, the red-favoring minority group was able to leverage their location advantage to persuade other agents to adopt red.

The consensus dynamics experiment raises several interesting questions about agent behavior, including:

- How should agents balance their efforts to promote their own preferred outcomes against the imperative to achieve some consensus?
- In pursuing their goals, how should agents take into account observed voting patterns of neighbors, and their partial knowledge of network structure?
- Given that each agent in the network only has a limited local view of others' actions, to what extent do network structure and payoff function affect agents' ability to reach consensus and subsequently determine consensus outcomes?

Alternatively, rather than ask how agents *should* behave, one could pose questions about

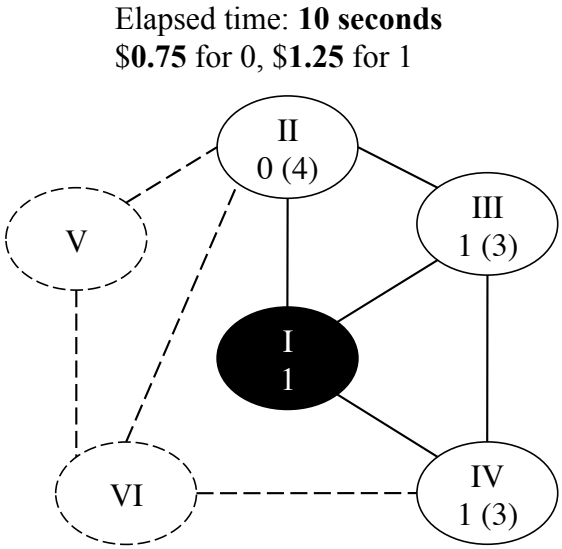

Figure 4.3: A typical game experiment from an agent's perspective (Kearns et al., 2009). From the view of agent I (shaded node) who currently picks action 1, agents II, III, and IV choose actions 0, 1, and 1, and have node degrees of 4, 3, and 3 (in parenthesis) respectively. It is also aware of the existence of II, III and III, IV edges. Agent I observes nothing about agents V and VI.

how agents *do* behave. One may be interested, for example, in the behavior of human agents, or of artificial agents constructed in various ways or induced by specified learning processes.

Kearns et al. (2009) conducted a series of human-subject experiments to collect data about how human agents behave in 9 different instances of the consensus dynamics game, each of which contains 9 experiment runs. By varying preference assignments (payoff function) and network structure, they gathered evidence about the effect of these factors on strategies employed, and the consequent voting results. In particular, they were interested in developing models that would predict whether a given scenario would be likely to converge to consensus, and if so, how fast and on which outcome. A simple multiplicative model that combines vote preferences with trends in neighbor actions can provide an effective probabilistic model of subject behavior (Kearns et al., 2009; Kearns and Wortman, 2008). In essence, this model is an individual behavior model that assumes conditional

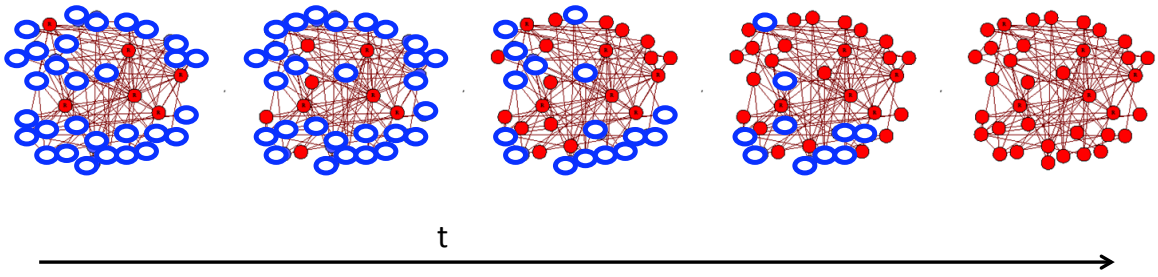

Figure 4.4: Time snapshots of a lab experiment run where the densely connected minority group (the filled red nodes in the leftmost snapshot) that preferred red exerted strong influences on the blue-favoring majority, which are generally much less connected. The minority group eventually succeeded in converting all the initial (unfilled) blue votes to (filled) red votes.

independence of agent behavior given history. As complexity limitations necessitate abstraction in consensus dynamics games of this size, explicit modeling of joint behavior using hGMMs offers potential gains in predictive power in this domain.

4.4 Modeling Behavior with Simulated Data

I next address the problem of predicting voting choices in the network consensus dynamic scenario described in Section 4.3. In particular, I demonstrate how to model behavior from a known generative model (the ground truth), which allows me to directly calibrate predictive accuracy in different experiment settings. The generative model adopts a version of the fictitious play process, where agents update their beliefs and choose their responses independently given history. This fictitious play process is itself an individual behavior model, thus suggesting that an individual multiagent behavior model could optimally capture the generated dynamic behaviors given full history. Therefore, any advantage a joint multiagent behavior model may have over an individual multiagent behavior model in this context, given abstracted history, can be argued to apply, all the more, for other cases where agents do not pick their actions independently. The focus of this study is on learning the

models' parameters, taking the graphical structure as a given input. The study starts with the assumption that agents pick actions synchronously, and later introduces a second, asynchronous scenario.

4.4.1 Model Specifications

I describe the specifics of a *joint behavior consensus model* (JCM) and an *individual behavior consensus model* (ICM) designed to capture agent decisions given data sampled from the generative model of smooth fictitious play described in Section 4.4.2 below. Note that the main goal here is to compare these two models by their abilities to capture conditionally independent agent behavior given abstracted history, but not to construct new improved analytic models for the consensus dynamics scenario.[3] Therefore, the generative model's function form is directly adopted in both JCM and ICM.

This study further assumes for the joint behavior hGMM that $N_i = \Gamma_i = N_i^{\mathsf{O}}$, which is given as input, for all i, and thus uses N_i as notation for all the neighbor sets. Recall that $I(x, y)$ is an indicator function: $I(x, y)$ equals 1 if $x = y$, and 0 otherwise. A history $H_{N_i}^t$ of length h is summarized by the frequencies of actions chosen in the neighborhood over the previous h time periods. Specifically, denote $f(a_i, H_{N_i}^t)$ as the frequency of action a_i being chosen by other agents in $H_{N_i}^t$,

$$f(a_i, H_{N_i}^t) = \frac{\sum_{k \in N_i - \{i\}} \sum_{\rho=t-h}^{t-1} I(a_i, a_k^\rho) + \epsilon}{h|N_i - \{i\}|}.$$

The frequency function f captures the degree to which a_i is similar to past choices by i's neighbors in $H_{N_i}^t$. Its joint analog reflects the historical frequency of a *local joint action* a_{N_i},

$$f(a_{N_i}, H_{N_i}^t) = \frac{\sum_{\rho=t-h}^{t-1} I(a_{N_i}, a_{N_i}^\rho) + \epsilon}{h}. \tag{4.4}$$

The term ϵ is added in both definitions above to ensure that the corresponding term does

[3] Section 5.2 discusses improvements on existing analytic models, given human-subject experiment data.

not vanish when the action a_i or the configuration a_{N_i}, respectively, does not appear in $H^t_{N_i}$. To simplify exposition, I henceforth drop the time superscript t and the dependence neighborhood subscript N_i from $H^t_{N_i}$, taking these modifiers of history H as understood. The JCM potential function is designed to capture the impact of past collective choices of i's dependence neighborhood, and i's relative preference for each action, as reflected in the reward function $r_i(a_i)$. JCM encodes other factors bearing on choice of action a_i by a parameter α_{i,a_i}.

The potential function for agent i is given by

$$\pi_i(a_{N_i} \mid H) = \exp\Big(\beta_i r_i(a_{N_i}) f(a_{N_i}, H) + \alpha_{i,a_i}\Big), \tag{4.5}$$

where for all i and $c \in A_i$, $0 \leq \alpha_{i,c}, \beta_i \leq 1$, and $\beta_i + \sum_c \alpha_{i,c} = 1$. The term $r_i(a_{N_i})$ is the expected reward for agent i given its dependence neighborhood's joint action a_{N_i}. If the strict definition of reward were adopted from the game description, $r_i(a_{N_i})$ would be non-zero only when all actions in a_{N_i} are the same. A modification is added to model the network contagion phenomenon:

$$r_i(a_{N_i}) = \gamma^{\sum_{k \in N_i}(1 - I(a_i, a_k))} r(a_i), \tag{4.6}$$

for $\gamma \in (0, 1]$.[4] Observe that $r_i(a_{N_i})$ as defined is increasing in the number of i's neighbors choosing a_i, reflecting the positive influence of neighbor choices on i.

A fair comparison between the two hGMMs, the joint behavior JCM and the individual behavior ICM, must seek to preserve as many parameters from the JCM representation as possible in constructing an ICM for the consensus dynamics scenario. I thus define the probabilistic ICM strategy as follows:

$$\Pr(a_i \mid H) = \frac{1}{Z_i} \exp\Big(\beta_i r_i(a_i) f(a_i, H) + \alpha_{i,a_i}\Big). \tag{4.7}$$

[4] I fix the value of γ at 0.9 in all studies of the consensus dynamics experiment in both Chapters IV and V.

As above, for all i and $c \in A_i$, $0 \leq \alpha_{i,c}, \beta_i \leq 1$, and $\beta_i + \sum_c \alpha_{i,c} = 1$, and Z_i is the normalization factor over all $a_i \in A_i$. Note that this normalization sums over only the action space of a single agent and is, therefore, inexpensive to compute. As a result, computing (exact) inferences on JCM is exponentially more expensive than on ICM, while the identical number of parameters in both models assures that they have the same degrees of freedom.

4.4.2 Smooth Fictitious Play Simulation

For a consensus dynamics scenario with a given observation graph, one can generate data using a *smooth* fictitious play process (Camerer and Ho, 1999). In fictitious play, each agent maintains probabilistic beliefs about the actions of other players, based on the observed frequency of history behavior. Here in the consensus dynamics scenario, each i maintains beliefs about only its neighbors N_i^{O}'s actions, which it supposedly can observe. Let $b_{ij}(a_j)$ be i's belief that j chooses action a_j. At the beginning of each simulated experiment run, each agent starts with uniform beliefs: $b_{ij}(a_j) = \frac{1}{|A_j|}$. Agents update their beliefs as follows. If at time t, j executed action a'_j, then for all a_j, and every neighbor i of j,

$$b_{ij}(a_j) \leftarrow \frac{b_{ij}(a_j)t + I(a_j, a'_j)}{t+1}.$$

Whereas in classic fictitious play agents respond optimally to their beliefs, in the smooth version agents select actions probabilistically in proportion to the expected reward computed with respect to these beliefs. I adopt an existing multiplicative model[5] for this game (Kearns et al., 2009), which weighs the probability that an agent takes action a_i in proportion to the product of its reward $r_i(a_i)$ and the probability that all neighbors play the same action, $\prod_{j \in N_i^{\mathsf{O}}} b_{ij}(a_i)$. Thus, in each round the simulation samples each agent i's action according to the probability distribution

[5]This also serves as the basis for the *proportional response model*, described in Section 5.2.

$$\Pr(a_i) \propto r_i(a_i) \prod_{j \in N_i^{\mathsf{O}}} b_{ij}(a_i). \tag{4.8}$$

The simulation terminates when all agents reach a consensus or the allotted time runs out. The fictitious play simulation qualifies as an individual behavior model because each agent chooses its action independently from others conditional on the commonly seen history. Moreover, this simulation model incorporates the full-length history in modeling and consequently generating agent actions.

4.4.3 Experiments

I empirically evaluate the predictive power of the two hGMMs, JCM and ICM, in the consensus dynamics scenario described in Section 4.4.1. The training data is generated by the smooth fictitious play process as described in Section 4.4.2.

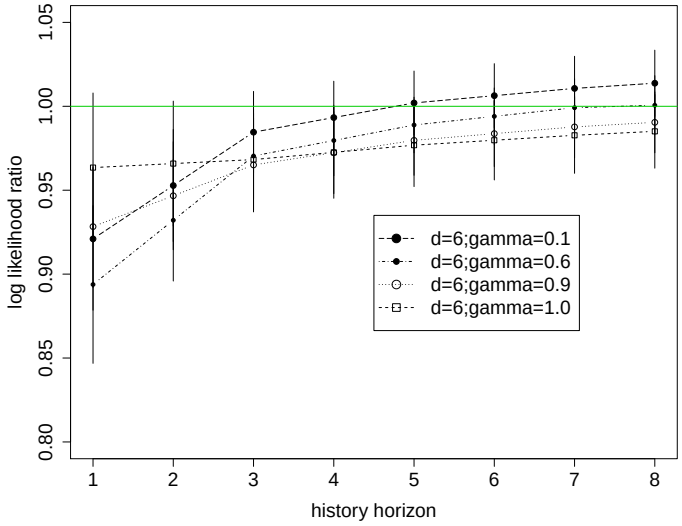

Figure 4.5: Predictive performance ratios R as a function of horizon. JCMs outperform ICMs ($R < 1$) across scenarios of varying γ.

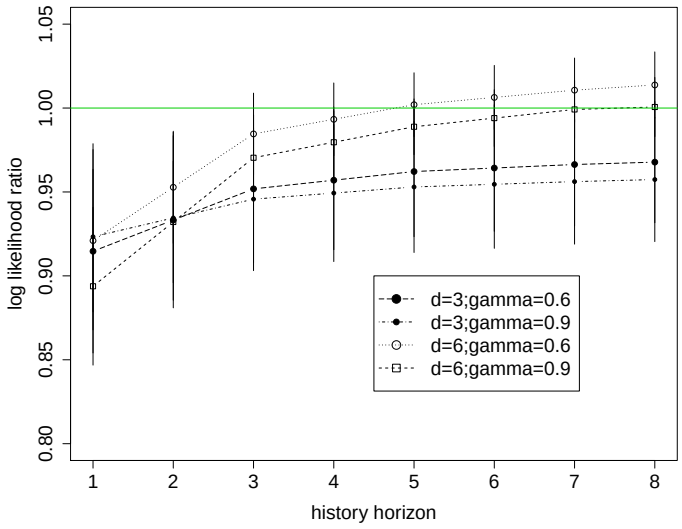

Figure 4.6: JCMs provide better predictions than ICMs across scenarios of varying maximum node degree d.

Experiment Settings

Consider consensus dynamics scenarios with 10 agents. A game instance specifies agent payoffs for each consensus outcome $\sim U[0, 1]$, as well as an observation graph with maximum degree controlled to be d. Each experiment provides results averaged over 20 game instances. A data set of 20 game runs is then generated for each game instance by simulating independent runs of the smooth fictitious play process defined in Section 4.4.2. Each simulation run lasts until either everyone agrees on one of the two voting options or the time limit T (set at 50) is reached. I use $d = 6$ and $\gamma = 0.9$ (see Section 4.4.1) in the simulations unless otherwise specified.

This study includes both synchronous and asynchronous game scenarios. In the synchronous scenario, agents update their beliefs and choose a response simultaneously in every round. The asynchronous version allows agents to change their votes at any time. The generative model described in Section 4.4.2 assumes a synchronous update model. In order to mimic asynchrony in voting decisions, I allow agents to have different update rates, with agent i updating its vote every t_i time periods. Specifically, every t_i is set to some value at the start of each simulation run. Whereas I implement this using discrete-

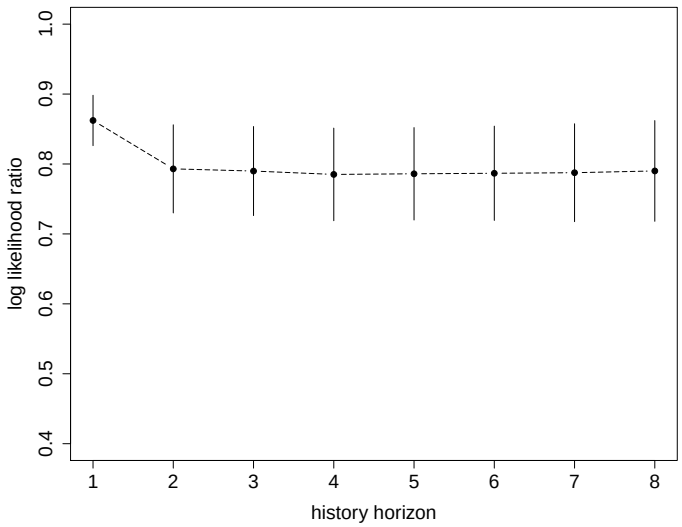

Figure 4.7: JCMs provide better predictions than fictitious play sampling.

event simulation, I achieve the effect of a more continuous decision process by aggregating actions within a *summarization interval*: a window of length v that combines all decisions v rounds at a time. Under this transformation, all actions that occur during the same epoch (that is, v simulated time steps) are grouped to be effectively simultaneous.

The learned dynamic multiagent models are evaluated by their ability to predict future outcomes, as represented by a testing set Y in the same format as the training set X. Given two models M_1 and M_2, one can compute the ratio of their corresponding log-likelihood measures for the testing data set Y: $R_{M_1/M_2}(Y) = \frac{L_{M_1}(Y)}{L_{M_2}(Y)}$. For each game instance, I train the models using 10 game runs, and compare their performance on the remaining 10 runs by the ratio $R_{\text{JCM/ICM}}$. Note that since log likelihood is negative (certain predictions are excluded), $R_{\text{JCM/ICM}} < 1$ indicates that JCM is better than ICM at predicting Y, and vice versa if the ratio exceeds one. Recall that all results present averages over the 20 game instances.

Results

In the first set of experiments I evaluate the performance of JCMs (as compared to ICMs) as functions of the history horizon length for values of $\gamma \in [0, 1]$. I display results

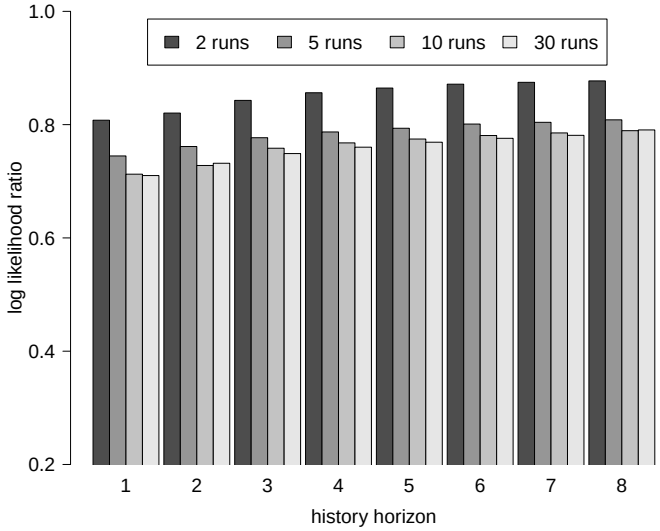

Figure 4.8: The effects of training data availability on the performance of JCMs in comparison with untrained JCMs.

for a representative selection, $\gamma \in \{0.1, 0.6, 0.9, 1.0\}$. The results in Figure 4.5 show that in general JCMs outperform ICMs in predicting agent behavior for shorter history horizons. The difference between these models' performance seems to vanish as the history horizon increases. Since agents condition their actions on the complete history in the generative fictitious play simulation, a history of limited horizon renders agent actions correlated from the modeler's perspective. As a result, JCMs, which directly model joint behavior, are more effective due to capturing action correlations induced by truncated history, even though the generative model is in fact an individual behavior model. Figure 4.5 also indicates that for shorter history lengths, the relative performance of JCMs peaks when γ is around 0.6, whereas for longer histories the joint model has greater advantage for higher γ. The reason is that action correlations are less prominent when history horizon increases, and, consequently, models closer to the generative individual behavior multiagent model—in this case, those with higher γ (equating the different reward terms $r_i(a_{N_i})$ and $r_i(a_i)$ per (4.6))—perform better.

To evaluate the effect of observation graph density on joint agent behavior, I vary the maximum degree d of generated graphs. The results in Figure 4.6 for $d = 3$ and $d = 6$

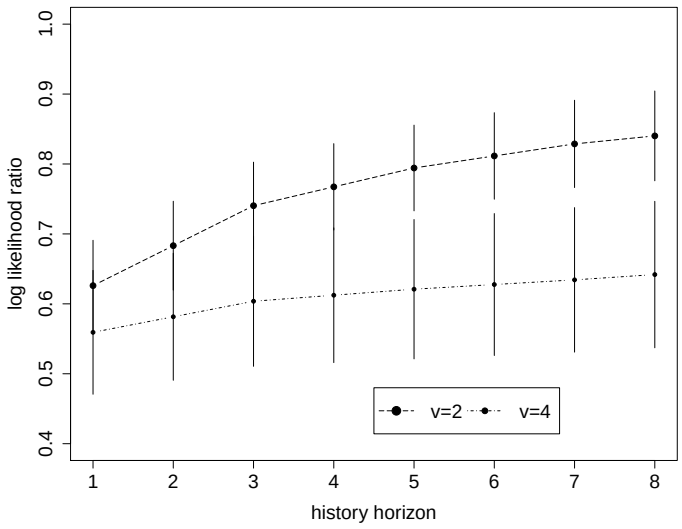

Figure 4.9: JCMs provide better predictions than ICMs in asynchronous scenarios, for two summarization intervals v.

suggest that JCMs are better on sparser graphical structures. The intuition behind this is that action configurations in smaller dependence neighborhoods tend to repeat more often and, consequently, increase the contribution of the $f(a_{N_i}, H_{N_i})$ term in the JCM potentials, enhancing their power in capturing joint behavior.[6]

A natural alternative to either JCMs or ICMs is *fictitious play sampling*, which effectively mimics how one would use a generative fictitious play model to predict future joint actions. Specifically, fictitious play sampling computes the probability of an action profile a given history H as the empirical distribution of a in training data conditional on H. The experimental results in Figure 4.7 show that JCMs consistently outperform fictitious play sampling, and this advantage is smallest at $h = 1$ and greatest at $h = 4$. Intuitively, shorter history horizons imply that more data from the training data set can be used in the fictitious play sampling model to compute $\Pr(a \mid H)$ as there are more instances of the specific history H for shorter horizon h, leading to more accurate predictions, even relative to JCMs. The main illustrative point of this experiment set is that direct sampling from a fictitious play model is not only more computationally expensive but also less powerful in predicting

[6]Of course, there is no difference between JCMs and ICMs to be found when $d = 0$ and, thus, this difference is likely to be small for *very* sparse graphs.

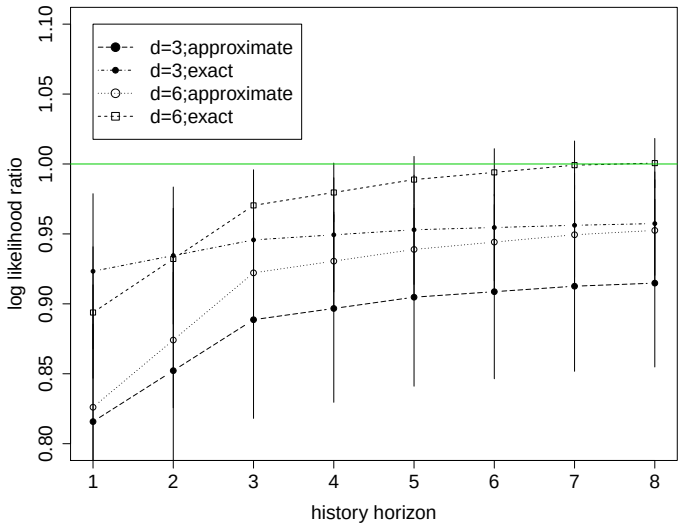

Figure 4.10: JCMs show greater predictive power than ICMs when employing generalized belief propagation approximation.

outcomes than a JCM capable of effectively extracting information about agent behavior from the same data set.

To assess the added value of parameter learning, I compare trained JCMs with untrained models, where parameter values are chosen uniformly randomly. Specifically, I used untrained JCMs (training data set size of 0) as a baseline, and varied the size of the training data in the set $\{2, 5, 10, 30\}$. The corresponding likelihood ratio of trained to untrained hGMMs evaluated on test data are displayed in Figure 4.8. In essence, the outcomes confirm that the default size for the training data set (10 game runs) is sufficient for learning JCMs. Analogous results were also obtained for ICMs.

The next set of experiments evaluates the impact of asynchrony in agent behavior on the relative efficacy of JCMs and ICMs. Specifically, consider two summarization intervals, $v = 2$ and $v = 4$, with the results shown in Figure 4.9. As expected, joint-behavior modeling is more advantageous for the longer summarization interval. Moreover, the predictive performance of JCMs relative to ICMs for both summarization window sizes here is greater than what is observed in the synchronous environment (which would roughly correspond to $v = 1$, although all agents also make their decisions simultaneously in that model). To

understand this phenomenon note that actions that occur in different periods that are collapsed to fall in the same summarization window v in an asynchronous scenario are treated as simultaneous actions in both models, even though earlier actions may well cause the later ones. As v increases, more cause-effect action pairs are grouped in the same rounds in the transformed (synchronized) data, yielding greater action correlations that are not modeled in ICMs. The results suggest that JCMs may be especially efficacious in asynchronous environments or when observations are necessarily aggregated in time intervals.

A real concern about using joint behavior hGMMs in realistic settings is the scalability of the modeling techniques when the number of agents is large. While all the problems considered here are small enough to enable exact learning, large systems would require one to use approximation in computing the normalization factor.

I next proceed to verify that approximation using generalized belief propagation is feasible in this setting and the advantage of joint behavior models over individual behavior models does not dissipate when learning is approximate rather than exact. Figure 4.10 illustrates the performance of JCMs that use both exact and approximate inference. In fact, approximate JCMs even outperform JCMs that use exact inference. My intuition for this surprising result is that action profiles with sufficiently small values of potentials are likely to be dropped from the approximate computation of Z. As a result, approximate JCMs become more focused in explaining and predicting more frequent outcomes, while getting punished more when a less frequent outcome occurs. More significantly, approximate JCMs outperform ICMs in predicting the game's sequential outcomes, allowing joint behavior hGMMs to scale to realistic scenarios. That approximate inference does not degrade the predictive power of hGMMs in this study is a promising indicator for application of hGMMs in large networks. This finding helps to promote usage of history-dependent graphical multiagent models in efficiently and effectively inferring about system dynamics given abstract representations of history that may induce action correlations from the modeler's perspective.

CHAPTER V

Learning Agent Dependence Structures

The graphical structures that capture various types of interactions and interdependencies among agents are unavailable or partially invisible to the modelers in many real-world scenarios. Hence, the problem of discovering these dependence structures is essential to many multiagent behavior modeling tasks.

In this chapter, I first address the problem of learning graphical game models that capture *payoff (inter)dependence* among agents, given observational data on their payoffs (Duong et al., 2009). In particular, I examine situations where there is some game which can be represented compactly by a graphical model, but the payoff functions of players are unknown. Given a data set of payoff realizations for specific strategy profiles, which are drawn according to some fixed probability distribution, I seek to learn the underlying graphical game model. For example, the game may be defined by a simulator (Wellman, 2006), from which one has a limited budget of payoff observations. The goal is to learn a graphical representation of the game based on this payoff experience.

Findings from the study on learning graphical games help to inform the solution to the chapter's other main problem: learning *hGMM (inter)dependence* structures from observational data of dynamic multiagent scenarios (Duong et al., 2012). In previous chapters I have presumed a fixed graph structure defined by the modeler, and subsequently constructed hGMMs whose interdependence graph is identical to the given structure. How-

ever, it is not always apparent how to choose the most salient inter-agent dependencies for accurate and tractable modeling, given only some or no dependence relationships as input. I propose a greedy algorithm for simultaneously learning the graphical structure and parameters of an hGMM that can effectively and compactly capture joint dynamic behavior. I then empirically evaluate the learning technique with data from human-subject laboratory experiments (Kearns et al., 2009) on the consensus dynamics scenario outlined in Section 4.3, illustrating the benefits of inducing interdependence structures from data in improving predictive performance.

5.1 Learning Graphical Game Models

Consider a modified version of the mayor election example originally introduced in Chapter I. In particular, each voter's utility is a function of how many people in the voter's social circle chooses the same candidate that he or she eventually votes for, as previously described in Section 3.1. The relationship graph essentially specifies whose actions are factored into each voter's utility function. Presumably, due to privacy reasons, May the political analyst does not have access to the relationship graph, and moreover collects data only on how optimistic each voter is about the current election, which approximates their utility. May is particularly interested in inducing the graph from the utility data she has collected, as the relationship graph itself is important to modeling multiagent behavior in this scenario.

In this section I examine settings where there is some game scenario represented by a graphical game model. As defined in Section 2.3, a graphical game model is a tuple $\mathcal{G} = (I, \{A_i\}, \{J_i\}, \{u_i(\cdot)\})$, where $I = \{1, \ldots, n\}$ is the set of players, and A_i is player i's strategy space for every $i \in I$. A strategy profile $a = (a_1, \ldots, a_n)$ represents the strategies of all players. Player i's payoff depends on only its own action and the actions of its neighbors in the set J_i: $u_i(a_i, a_{J_i}) : A_i \times A_{J_i} \rightarrow \mathbb{R}$. The notation $a_{J_i} \subset a_{-i}$ means that each $j \in J_i$ plays its assigned strategy from a_{-i}. The game structure is captured by a graph

(V, E) in which $V = I$ and there is an edge $e = (j, i) \in E$ if and only if $j \in J_i$ (directed edge) or $j \in J_i \wedge i \in J_j$ (undirected). Recall that the input in this problem is a data set of payoff realizations for specific strategy profiles, which are drawn according to some fixed probability distribution from the game's payoff function, which is unknown.

Learning graphical structures has been explored in many other contexts, notably that of inducing Bayesian network structure from data. The general version of this problem is NP-hard, and so heuristics are commonly employed, particularly variants of local search (Heckerman et al., 1995; Friedman, 1998; Jensen and Nielsen, 2007).[1] Inducing a compact representation of a joint distribution over random variables is generally formulated as an unsupervised learning problem, since the given input consists of data reflecting the distribution rather than a label specifying the distribution itself. In contrast, the problem of inducing a compact representation of a payoff function given a sample of action profile and payoff tuples falls in the supervised learning paradigm. A similar direction was pursued by Ficici et al. (2008), who learn a cluster-based representation of games that takes advantage of game-theoretic symmetries wherever these exist. Most recently, Gao and Pfeffer (2010) have expanded the game structure learning problem formulated and presented in this chapter (Duong et al., 2009) to encompass both game structure and player strategies. In particular, they assume that players generally play reasonably good strategies. This assumption generates additional constraints to the learning problem, allowing them to improve the accuracy of the learned games.

The main contributions in this graphical game study are to formally formulate the problem of learning graphical game models, and to introduce metrics for evaluating different learning algorithms. In particular, given the varied goals of graphical game learning, my evaluation metrics correspond to each of three possible aims: (i) learning the actual graphical structure (e.g., social network) among the players, (ii) computing game-theoretic solutions, and (iii) learning the payoff functions to better understand structural properties of the

[1] Schmidt et al. (2007) present an alternative heuristic method based on L_1-regularized regression.

game (perhaps with the purpose of developing better and simpler stylized models). I then present four algorithms for learning graphical game structures from payoff data, which are basically derivations of well established structural learning methods, put in the new context of graphical game models. The focus here is not these algorithms per se, but the examination of their performance in learning graphical game structures, as measured by the three proposed evaluation metrics.

5.1.1 Problem Definition

Suppose that the given input includes only players and action sets $[I, A]$, but not payoff functions. I refer to such a specification as a *game form*. The input additionally includes a data set $D = \{(a^1, U^1), \ldots, (a^m, U^m)\}$ of profiles a^k and corresponding payoffs U^k (or noisy samples from the true payoff function). Let n be the number of agents: $n = |I|$. Also define $U^l = (U_1^l, \ldots, U_n^l)$ for each strategy profile a^l, where U_i^l specifies the payoff for player i, when all players choose a. The goal is to induce the graphical game structure, or more specifically, to seek an algorithm that takes as input data set D and game form $[I, A]$, and outputs a graphical game $\hat{\mathcal{G}} = [I, \{A_i\}, \{\hat{J}_i\}, \{\hat{u}_i(\cdot)\}]$ with $\hat{J}_i$ the estimated collection of neighbors of player i and $\hat{u}_i(\cdot)$, i's estimated utility function.

One of the basic ways to design and evaluate learning algorithms is by defining a *loss function*. Given true payoffs u, I define the *true* (quadratic) loss of a learned graphical game $\hat{\mathcal{G}}$ for player i to be

$$\Omega_i(\hat{J}_i, \hat{u}_i) = E_{a \sim P}[(u_i(a) - \hat{u}_i(a_i, a_{\hat{J}_i}))^2], \tag{5.1}$$

where P is some distribution over profiles,[2] and $a_{\hat{J}_i}$ is the vector of actions played by the players in $\hat{J}_i$. Whereas the true loss is a measure of learning performance, learning algorithms are typically designed to minimize *empirical* loss. Define the empirical loss for

[2] In the evaluation section I assume a uniform distribution.

player i, given data set D, to be

$$\hat{\Omega}_i(\hat{J}_i, \hat{u}_i) = \frac{1}{m}\sum_{d=1}^{m}(U_i^d - \hat{u}_i(a_i^d, a_{\hat{J}_i}^d))^2. \tag{5.2}$$

Given the definition of loss for a particular player, define the true loss of a learned graphical game $\hat{\mathcal{G}}$ by

$$\Omega(\{\hat{J}_i\}, \hat{u}) = \frac{1}{n}\sum_i \Omega_i(\hat{J}_i, \hat{u}_i). \tag{5.3}$$

The empirical loss is defined analogously.

The loss functions above are defined given the utility estimates $\hat{u}_i(\cdot)$. A natural estimator of $\hat{u}_i$ at $(a_i, a_{\hat{J}_i})$ is a sample mean over all profiles in D consistent with that partial profile:

$$\hat{u}_i(a_i, a_{\hat{J}_i}) = \frac{1}{|D_{(a_i, a_{\hat{J}_i})}|}\sum_{(a^l, U^l)\in D_{(a_i, a_{\hat{J}_i})}} U_i^l, \tag{5.4}$$

where $D_{(a_i, a_{\hat{J}_i})}$ denotes the set $\{(a^l, U^l) \in D \mid a_i = a_i^l, a_{\hat{J}_i} \subset a_{-i}^l\}$.[3] To see that this is a sensible utility estimator, note in the following observation that it has a natural invariance property with respect to edges not present in the actual game graph.

Observation V.1. Suppose that D contains all profiles $a \in A$, and payoffs are obtained with no noise. For any i, $\hat{J}_i$, and $j \notin J_i$, $\hat{u}_i(a_i, a_{\hat{J}_i}) = \hat{u}_i(a_i, a_{\hat{J}_i'})$ for all $a \in A$, where $\hat{J}_i' = \hat{J}_i \cup j$.

Henceforth, assume that $\hat{u}$ is obtained using (5.4) and use the shorthand notation $\hat{\Omega}_i(\hat{J}_i)$, or simply $\hat{\Omega}_i$.

Definition V.2. The empirical loss function $\hat{\Omega}_i$ is *monotone* if $\hat{J}_i \subseteq \hat{J}_i'$ implies that $\hat{\Omega}_i(\hat{J}_i) \geq \hat{\Omega}_i(\hat{J}_i')$.

Theorem V.3. *Let $\hat{\Omega}_i$ and $\hat{u}_i$ be as defined in* (5.2) *and* (5.4) *respectively. Then for any $\hat{J}_i$, j, $\hat{\Omega}_i(\hat{J}_i) \geq \hat{\Omega}_i(\hat{J}_i \cup j)$.*

[3] Observe that this estimator is undefined when $|D_{(a_i, a_{\hat{J}_i})}| = 0$. I address this issue in Section 5.1.4 and choose to ignore it for now.

Proof. To begin, rewrite the loss function using somewhat different notation given an arbitrary neighborhood J:

$$\hat{\Omega}_i(J) = \frac{1}{n}\sum_{l=1}^{n}(U_i^l - \hat{u}_i(a_i^l, a_J^l))^2 = \frac{1}{n}\sum_{a_i \in A_i}\sum_{a_J \in A_J}\sum_{a' \in D_{(a_i, a_J)}}(U_i(a') - \hat{u}_i(a_i', a_J'))^2,$$

where $U_i(a')$ denotes the utility corresponding to the data point in D with the strategy profile a'.

Now fix a_i, a_j, a_J and note from Equation (5.4) that given fixed a_i, a_j, a_J, both $\hat{u}_i(a_i, a_J)$ and $\hat{u}_i(a_i, a_J, a_j)$ are constant. Define

$$\hat{\Omega}_i(J, a_i, a_j, a_J) = \sum_{a' \in D_{(a_i, a_j, a_J)}}(U_i(a') - \hat{u}_i(a_i', a_J'))^2 \tag{5.5}$$

and define $\hat{\Omega}_i(J', a_i, a_j, a_J)$ analogously. Now define

$$\Delta\hat{\Omega}_i(a_i, a_j, a_J) = \hat{\Omega}_i(J, a_i, a_j, a_J) - \hat{\Omega}_i(J', a_i, a_j, a_J).$$

Then by using (5.5) one can have

$$\begin{aligned}
\Delta\hat{\Omega}_i(a_i, a_j, a_J) &= \sum_{a' \in D_{(a_i, a_j, a_J)}}\left[(U_i(a') - \hat{u}_i(a_i', a_J'))^2 - (U_i(a') - \hat{u}_i(a_i', a_{J'}'))^2\right] \\
&= \sum_{a' \in D_{(a_i, a_j, a_J)}}[(U_i(a') - \hat{u}_i(a_i, a_J) - U_i(a') + \hat{u}_i(a_i, a_j, a_J)) \\
&\qquad (U_i(a') - \hat{u}_i(a_i, a_J) + U_i(a') - \hat{u}_i(a_i, a_j, a_J))] \\
&= \sum_{a' \in D_{(a_i, a_j, a_J)}}[\hat{u}_i(a_i, a_j, a_J) - \hat{u}_i(a_i, a_J)) \\
&\qquad (2U_i(a') - \hat{u}_i(a_i, a_J) - \hat{u}_i(a_i, a_j, a_J))]\,.
\end{aligned}$$

Now, since both $\hat{u}_i(a_i, a_J)$ and $\hat{u}_i(a_i, a_J, a_j)$ are constant, one can simplify the above ex-

pression by defining

$$\Delta\hat{u}_i = \hat{u}_i(a_i, a_J, a_j) - \hat{u}_i(a_i, a_J). \tag{5.6}$$

Plugging this in and taking the constant term $\Delta\hat{u}_i$ out of the sum to get

$$\begin{aligned}
\Delta\hat{\Omega}_i(a_i, a_j, a_J) = & \sum_{a' \in D_{(a_i,a_j,a_J)}} \Delta\hat{u}_i[2U_i(a') - \hat{u}_i(a_i, a_J) - \hat{u}_i(a_i, a_j, a_J)] \\
= & [2\Delta\hat{u}_i \sum_{a' \in D_{(a_i,a_j,a_J)}} U_i(a')] - [\Delta\hat{u}_i \sum_{a' \in D_{(a_i,a_j,a_J)}} \hat{u}_i(a_i, a_J)] - \\
& [\Delta\hat{u}_i \sum_{a' \in D_{(a_i,a_j,a_J)}} \hat{u}_i(a_i, a_j, a_J)] \\
= & 2\Delta\hat{u}_i|D_{(a_i,a_j,a_J)}|\hat{u}_i(a_i, a_j, a_J) - \Delta\hat{u}_i|D_{(a_i,a_j,a_J)}|\hat{u}_i(a_i, a_J) - \\
& \Delta\hat{u}_i|D_{(a_i,a_j,a_J)}|\hat{u}_i(a_i, a_j, a_J) \qquad (5.7) \\
= & \Delta\hat{u}_i|D_{(a_i,a_j,a_J)}|\hat{u}_i(a_i, a_j, a_J) - \Delta\hat{u}_i|D_{(a_i,a_j,a_J)}|\hat{u}_i(a_i, a_J) \\
= & \Delta\hat{u}_i|D_{(a_i,a_j,a_J)}|(\hat{u}_i(a_i, a_j, a_J) - \hat{u}_i(a_i, a_J)) \\
= & |D_{(a_i,a_j,a_J)}|\Delta\hat{u}_i^2 \geq 0. \qquad (5.8)
\end{aligned}$$

The third equality (5.7) follows from definition (5.4) and the observation that $\hat{u}_i(a_i, a_J)$ and $\hat{u}_i(a_i, a_j, a_J)$ are constant, while the last equality (5.8) is derived from definition (5.6).

□

Theorem V.3 establishes the following characteristic of the loss function:

Corollary V.4. *The empirical loss function is monotone with respect to the number of graph edges.*

This result allows one to establish that adding one player j to the neighbor of player i, that is adding one edge to the graph, does not increase empirical loss. Note that neither this monotonicity nor Observation V.1 needs to hold under alternate definitions of approximate payoff different from (5.4).

5.1.2 Constrained Loss Minimization

Given monotonicity, minimizing empirical loss is trivial unless there are some constraints on graph complexity. To control complexity, one can impose degree bounds M_i on the number of neighbors that a player i can have. The problem (for the directed case) becomes

$$\min_{\{\hat{J}_i\}_{i \in I}} \sum_i \hat{\Omega}_i(\hat{J}_i) \quad \text{s.t.} \quad |\hat{J}_i| \leq M_i \; \forall \; i \in I.$$

Alternative complexity constraints, such as on the total number of edges, tree-width, or other measure, could be incorporated in a similar manner (with algorithmic implications). Moreover, that the learned graph structure's complexity dictates the efficiency of computing game-theoretic solution concepts for the underlying game (Daskalakis and Papadimitriou, 2006; Kakade et al., 2003) also has an influence on the choice of an appropriate complexity constraint. Learning undirected graphs entails including an additional symmetry constraint, $j \in \hat{J}_i$ if and only if $i \in \hat{J}_j$. In the directed case, since the objective is additive in i and the neighbor constraints are independent, one can decompose the problem into n separate minimization problems.

5.1.3 Learning Algorithms

I present four algorithms for learning the graphical structure and payoff function of a game from data (Duong et al., 2009). The descriptions below cover directed graphical games; extension to the undirected case is conceptually straightforward.[4] The first algorithm is a complete branch-and-bound search in graph space, and the remaining three are heuristic methods: greedy edge selection and two versions of stochastic local search. Since the focus is on the decomposable problem of learning directed graphical games with degree constraints, I describe each algorithm in terms of learning the neighborhood structure for a

[4]Note that payoff dependencies in graphical game models are conceptually different from probabilistic dependencies in GMMs, resulting in different algorithmic implications. Therefore, I need not restrict the scope of this study to only undirected edges; in fact I choose to focus on directed graphs for simplicity reasons.

fixed player i.

Branch-and-Bound Search

The branch-and-bound search algorithm, BB, constructs a search tree in which each node corresponds to a partial specification of $\hat{J}_i$, the neighborhood of i. Formally, each node at depth h is associated with a bit vector, $b = \langle b_1, \ldots, b_h \rangle$, with $b_j = 1$ representing an assignment of j to the neighborhood, $j \in \hat{J}_i$. The root (depth 0) of the search tree represents the null assignment. The two children of the root node expand the vector to one element, b_1, the left child assigning $b_1 = 0$ and the right $b_1 = 1$. Similarly, at depth h, the left child expands the assignment with $b_h = 0$ and the right with $b_h = 1$. The leaves (at depth $n - 1$) of the search tree represent the 2^{n-1} possible vectors that fully specify player i's neighborhood.

The key to a branch-and-bound search is to provide upper and lower bounds on the loss function at every node. Let $\langle b_1, \ldots, b_h \rangle$ be a partial assignment at depth h. Monotonicity allows one to establish an upper bound by setting all $b_{h'}$, $h' > h$, to 0. This bound can be improved by sequentially adding as many edges as the problem's constraints allow. To obtain a simple lower bound, one can again take advantage of the monotonicity of empirical loss by setting $b_{h'}$, $h' > h$, to 1. One can improve on this by subsequently removing a single edge (j^*, i) with smallest $\Delta_{\hat{J}_i, j^*} \hat{\Omega}_i = \hat{\Omega}_i(\hat{J}_i) - \hat{\Omega}_i(\hat{J}_i \cup j^*)$, that is, with smallest impact on empirical loss.

A subtree at a node is pruned if the lower bound on loss is strictly greater than the upper bound at *some* other node. In addition to pruning the tree based on upper and lower bounds, one can also naturally prune subtrees due to the complexity constraints: for example, once $|\hat{J}_i| = M_i$, the remaining unassigned bits must be zero. Furthermore, monotonicity allows one to prune a subtree at level h if $\sum_{k=1}^{h} b_k + n - 1 - h < M_i$ (i.e., even adding every remaining edge would not reach the neighborhood constraint). One can implement the search as the standard breadth- or depth-first search, both of which would be complete

with the branch-and-bound, since there is only finite depth. Branch-and-bound together with either breadth-first search or depth-first search is complete. Literature on structure learning suggests that complete search will be intractable in many realistic settings, as the computational complexity is $O(2^n)$. Thus, I next introduce efficient heuristic techniques that may generate suboptimal but reasonable solutions.

Greedy Loss Minimization

The greedy algorithm selects among potential edges (j, i) based on the amount of loss reduction due to introducing the edge, given the edges added so far. That is, given a current configuration $\hat{J}_i$, the algorithm adds an edge from j to i that maximizes $\Delta_{\hat{J}_i,j}\hat{\Omega}_i$. Specifically, it adds one edge at a time until $|\hat{J}_i| = M_i$.

Local Search

I explore two versions of local search for loss-minimizing neighborhoods. Let $b_t = \langle b_{1,t}, \ldots, b_{n-1,t} \rangle$ be a vector in iteration t indicating which nodes are included in the neighborhood of i and suppose without loss of generality that $\sum_{i=1}^{n-1} b_{i,t} \leq M_i$ (otherwise edges can be removed arbitrarily to satisfy the constraint). Let $\hat{J}_{i,t}$ be the corresponding estimated set of neighbors of i and $\hat{\Omega}_{i,t} = \hat{\Omega}_i(\hat{J}_{i,t})$ its corresponding empirical loss.

The first search method ($SA\text{-}Simple$) uses a standard simulated annealing procedure to obtain the next iterate b_{t+1}. It generates a candidate b' by uniformly selecting among the vectors differing from b_t by one bit, and satisfying the degree constraint. The next iterate b_{t+1} is then chosen to be b' with probability $p_t(\hat{\Omega}_{i,t}, \hat{\Omega}'_i)$, and b_t otherwise, with

$$p_t(\hat{\Omega}_{i,t}, \hat{\Omega}'_i) = \begin{cases} \exp\left[-\frac{\hat{\Omega}'_i - \hat{\Omega}_{i,t}}{T_k}\right] & \text{if } \hat{\Omega}'_i > \hat{\Omega}_{i,t} \\ 1 & \text{otherwise,} \end{cases}$$

where T_k is a schedule of "temperatures" that govern the probability with which inferior

candidates are explored.

The second local search method ($SA\text{-}Advanced$) is also based on simulated annealing. However, instead of randomly selecting a single candidate b', it examines all (feasible) neighbors of b_t and associates each neighbor $b_{j,t}$ with a probability measure:

$$p_{j,t}(\hat{\Omega}_{i,t}, \hat{\Omega}_{j,t}) \propto \begin{cases} \exp\left[(\hat{\Omega}_{i,t} - \hat{\Omega}_{j,t})T_0\right] & \text{if } \hat{\Omega}_{j,t} \leq \hat{\Omega}_{i,t} \\ \exp\left[\frac{\hat{\Omega}_{i,t} - \hat{\Omega}_{j,t}}{T_k}\right] & \text{if } \hat{\Omega}_{j,t} > \hat{\Omega}_{i,t} \;\&\; \\ & \nexists c : \hat{\Omega}_{c,t} \leq \hat{\Omega}_{i,t} \\ 0 & \text{otherwise,} \end{cases}$$

where T_0 is the initial value of the declining temperature schedule T_k. Each candidate $b_{j,t}$ is chosen as the next iterate b_{t+1} with probability $p_{j,t}$. Note that as T_0 goes to infinity, this version of local search approaches the behavior of the greedy algorithm. However, the ability to bias the choice of next configuration in favor of greater loss-reducing candidates comes at the increased cost of computing empirical loss for all candidates differing on one edge.

5.1.4 Evaluation

I evaluate the graphical-game learning methods with computational experiments on randomly generated games. I begin with the case where the data set includes all profiles $a \in A$, and each player's utility is sampled with no noise. I then evaluate the algorithms in more interesting and realistic instances where only a subset of profiles have been sampled and provided as training data or where payoff observations are noisy.

These experiments are mainly concerned with directed graphical games, though I also present some results for small undirected games. Note that learning the graphical representation may be useful even when data are available to completely describe the game. A graphical model can considerably speed up equilibrium computation or other game-

theoretic analysis. For example, one can map a graphical game to a Markov random field that enables efficient computation of the network's maximum a posterior estimation configuration. This reduction effectively allows one to compute the original game's PSNE in polynomial time, instead of exponential time (Daskalakis and Papadimitriou, 2006). One may also wish to understand the graphical structure for its own sake, for instance, in analysis of social network structure. These goals are complementary to the underlying problem of learning the graphical model by minimizing measurable loss. I evaluate the aforementioned learning methods using metrics that correspond to the three central goals of learning graphical games.

Approximating True Loss

The first and most natural metric is the one used to set up the problem: approximating the true loss, as defined in (5.1). This is a typical goal of machine learning in any domain, although perhaps the least important in this case. Rather, I use a standard learning setup as a means to an end, where the end in this case is, perhaps, better captured by the two metrics that follow.

Approximating Graphical Structure

The second metric focuses exclusively on the graphical structure of the learned games. Given the graphs of the underlying game $G = (V, E)$, and the learned game $\hat{G} = (V, \hat{E})$, I define their *structural similarity* as:

$$\frac{|E \cap \hat{E}|}{|E|}.$$

Structural similarity measures how much the learned graph $\hat{G}$ resembles the underlying graph G. When all profiles $a \in A$ are included in the input data set and the payoff for each is exact, the proposed greedy heuristic is in fact optimal according to this metric for learning directed graphs. To see this, note that a direct consequence of Observation V.1 is that adding an edge (j, i) which is not in the actual graph never decreases empirical loss. As

a result, the greedy heuristic will only select edges which are in the underlying graph, until these are exhausted (i.e., until there is no other edge that decreases empirical loss). Thus, $\hat{E} \subset E$ and, hence, $E \cap \hat{E} = \hat{E}$. When one learns graphical games, $|\hat{E}| = \sum_i \min(M_i, |E_i|)$ both for the greedy heuristic and for any optimal algorithm, where $E_i = \{(j, i) \mid j \in J_i\}$. Given the graphical learning literature's context, this result is surprising even in the limited sense in which it holds. In the undirected case, this argument no longer applies even if the conditions of Observation V.1 are satisfied.

Approximating Nash Equilibria

I adopt the notion of regret as the basis for the third performance metric. Recall that for a strategy profile a, player i's regret is the maximum gain that player i could achieve by deviating from action a_i. The regret $\epsilon(a)$ of a strategy profile a is defined as the maximum regret among all players: $\epsilon(a) = \max_{i \in I} \epsilon_i(a)$. Let $\mathcal{G}$ be the actual game. Suppose that $\hat{Q}$ is a set of pure-strategy Nash equilibria of the learned graphical game $\hat{\mathcal{G}}$. In cases where there exist no exact pure equilibria, $\hat{Q}$ would comprise the profiles with smallest regret. Define the regret-based metric to be the average regret of all minimum-regret profiles in $\hat{Q}$ with respect to $\mathcal{G}$:

$$\frac{\sum_{a \in \hat{Q}} \epsilon(a)}{|\hat{Q}|}.$$

5.1.5 Empirical Results

I generated game instances for the experiments using the GAMUT game generation engine (Nudelman et al., 2004). Specifically, I investigated three graphical structures: 10-player random graphs with half of the edges of a fully connected graph, 13-player tree games with maximum degree 4, and 10-player star games. I sampled 25 random game instances of each type of graphical game, with payoffs sampled uniformly randomly on $[0, 1]$. It follows that each data point in the outcome is averaged over 25 learning instances. I considered games where players had two actions each.

I evaluate my algorithms in three settings. In the first, I only observe exact payoffs for every strategy profile in the game, and I am merely interested in learning a compact representation. The second setting is the same, except that payoffs are sampled with additive zero-mean normal noise. In the final setting only a small subset of strategy profiles is available for learning. Since the results for star games are similar to those for tree games, I discuss outcomes presented only for tree games henceforth.

Complete Game Data

I first consider a setting in which the data set includes exact payoff realization for every $a \in A$. The results for learning a directed graphical model on such data are presented in Figures 5.1 and 5.2. As one can readily observe, the greedy algorithm (GR) is nearly optimal and considerably better than other heuristics, a result which is robust across the metrics and game classes considered here. Furthermore, Figure 5.6 shows that GR has a substantially lower running time than BB, and is faster than $SA\text{-}Advanced$, the most competitive heuristic. Above I have already noted that in this setting GR is in fact optimal vis-a-vis the structural similarity metric, and so it comes as little surprise that it is also quite good on other metrics. Note that since GR is actually a special case of $SA\text{-}Advanced$, the parameters of $SA\text{-}Advanced$ are clearly set suboptimally. However, the point is that $SA\text{-}Advanced$ is a more complex algorithm that requires a significant amount of tuning for its several parameters, and here the added complexity does not pay off. Nevertheless, $SA\text{-}Advanced$ does outperform the simpler and faster $SA\text{-}Simple$.

I next consider undirected graphical games with five players. The additional symmetry constraint in learning undirected graphs prohibits the decomposition of the learning problem into smaller optimization tasks for each i. Therefore, the reduction in number of players was necessary to accommodate the increased expense of branch-and-bound search. One can observe that GR is now suboptimal in the structural similarity metric. However, across all three metrics, GR is still nearly as good as BB, and outperforms the other ap-

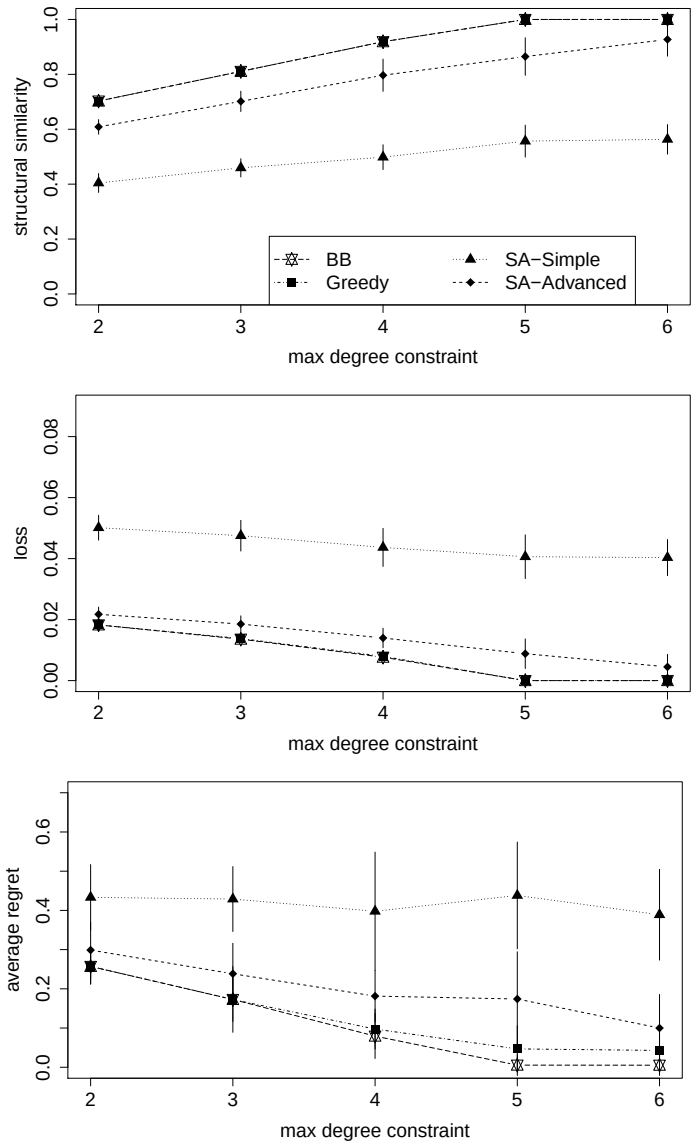

Figure 5.1: The greedy algorithm GR outperforms other heuristics and comes close to the optimal algorithm BB in learning directed-tree games from complete game data.

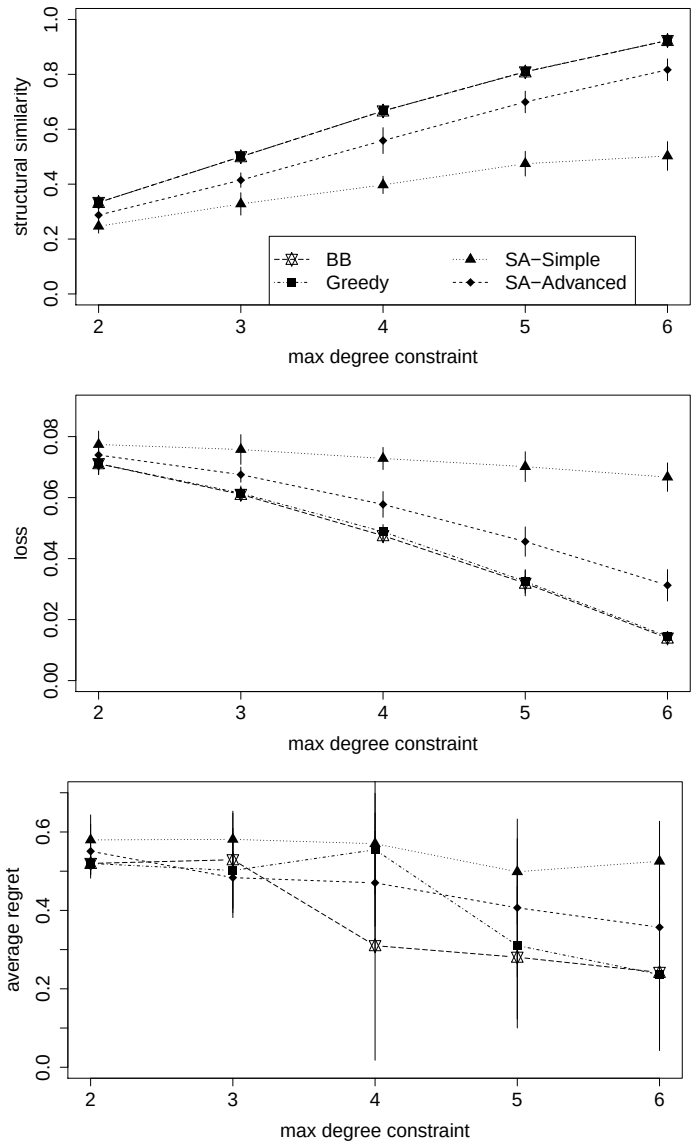

Figure 5.2: The greedy algorithm GR outperforms other heuristics and comes close to the optimal algorithm BB in learning random-graph games from complete game data.

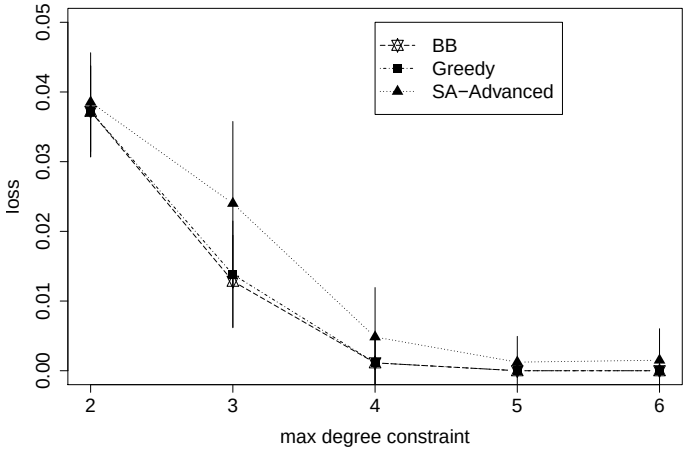

Figure 5.3: GR shows similarly good results (loss metric) in learning undirected random-graph games learned from complete game data.

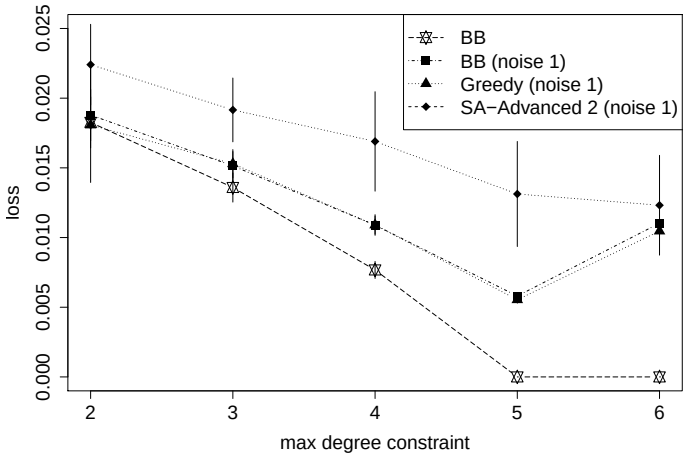

Figure 5.4: Loss when the entire game is sampled and added noise for the tree graph cases. The examined algorithms are robust to uniformly added noise.

proximate approaches, although its advantage is somewhat smaller here. A comparison of the methods on the loss metric is shown in Figure 5.3.

Finally, I consider tree games in which payoffs are generated with additive noise distributed according to $N(0, 1)$. In Figure 5.4 I plot the results for only the loss metric; the results for other metrics are qualitatively similar. One can see that the relative ranking of the methods is unchanged: GR is still nearly as good as BB, whereas $SA\text{-}Advanced$ tends to be significantly worse. I suspect that a part of the reason that GR appears so robust to noise is that the actual payoffs are uniformly randomly distributed, so $N(0, 1)$ noise does

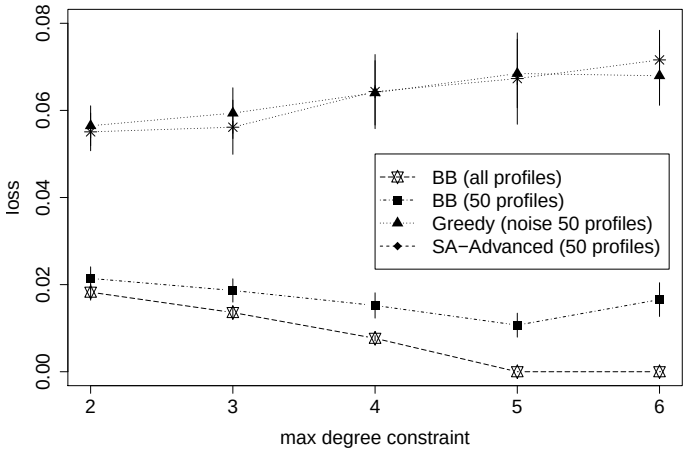

Figure 5.5: The optimal algorithm BB outperforms all heuristics substantially when only 50 profiles (out of 1024) are sampled and used in learning tree graphical games.

not significantly distort observations on average. More studies are needed to evaluate these learning algorithms with payoff noise drawn from different distributions.

Partial Game Data

I now evaluate the performance of the above learning algorithms when only a small fraction of strategy profiles have been sampled, assuming that the exact payoff vector is obtained for each. Another problem arising in this context is how to estimate payoffs for profiles in the learned graphical model for which no data is available. I address this issue here by using an average over the payoffs for all observed profiles.[5] In Figure 5.5, I present performance results for learning methods when only 50 profiles are observed. These 50 profiles are drawn uniformly randomly from the set of all profiles A ($|A| = 2^n$).

When the size of the data set is much smaller than game size, BB performs substantially better than GR (this is true for all three metrics; I show only the results for actual loss). Additionally, the advantage of GR over $SA\text{-}Advanced$ dissipates. As one can observe from Figure 5.6, there is no longer a significant speedup offered by GR over BB. Since Observation V.1 does not hold for the case of partial game data, both GR and BB may now

[5] I also tried other techniques, but the results seem to vary little.

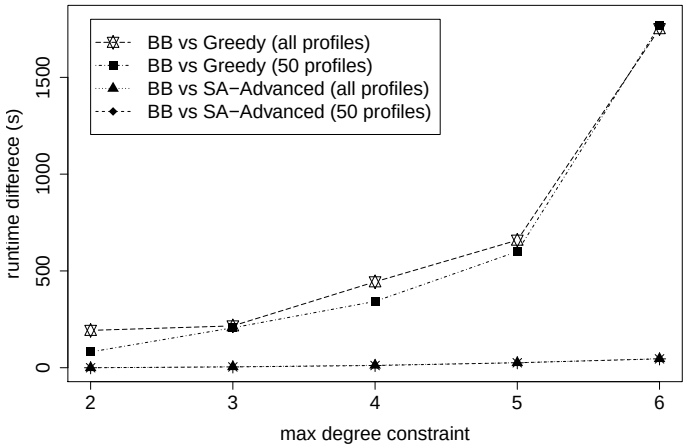

Figure 5.6: Runtime differences between branch-and-bound and approximate algorithms to learn tree graphical games. With a large training data set (all profiles), GR is faster than $SA\text{-}Advanced$, and substantially faster than BB. This speed up largely vanishes when given less training data (50 profiles).

include edges not present in the actual graph. Thus, the order in which GR adds edges becomes especially significant here, which, together with its policy of permanently adding edges, yields empirical loss well above optimal and above BB's.

These findings suggest that when a small subset of the profiles in the game is available, one should employ branch-and-bound (BB), as it considerably outperforms other methods and has competitive running time.[6] On the other hand, as the size of the data set becomes large, the advantage of BB over the greedy algorithm dissipates while its computation time increases considerably, particularly in learning undirected graphs. The results are similar when noise is added to payoff realizations.

5.2 Learning History-Dependent Multiagent Model Graph Structures for Predicting Dynamic Networked Behavior

The algorithms for learning payoff dependence structures of graphical game models in Section 5.1 provide valuable insights about the problem of learning probabilistic depen-

[6]One may observe that pruning generates better than a factor of two reduction in BB running time.

dence structures for dynamic multiagent scenarios. In particular, this section focuses on similar algorithms for inducing dependence structures of history-dependent GMMs from time-series observations of agent actions. Note that the interaction graphical structure that captures inter-agent communication is presumably given as input, and thus requires no learning effort. The main objective instead is to demonstrate the benefits of constructing probabilistic dependence structures, different from the given interaction structures, for modeling dynamic networked behavior in the example scenario of consensus dynamics.

The dependence graph structure of the optimal predictive hGMM need not mirror the interaction graph structure of a multiagent scenario. Furthermore, many complex multiagent scenarios render computation on the corresponding hGMMs intractable. Therefore, I propose a greedy algorithm to learn the graphical structure and parameters of an hGMM that can effectively and compactly capture joint dynamic behavior.

I employ human-subject experiment data in learning hGMMs and evaluating the learned models' predictions on voting behavior by comparing their performance with those of different baseline multiagent models. The results generally reveal that models expressing joint behavior outperform the alternatives, including models originally proposed by authors of the consensus dynamics experiments, in predicting voting dynamics. The joint behavior model provides comparable predictions on the rate of reaching consensus, and superior predictions of which consensus is reached. I further examine the learned hGMM graphical structures in order to gain insights about the dependencies driving voting behavior, as well as the network structure's effect on collective actions.

5.2.1 Model Specifications

The joint behavior hGMM of consensus dynamics developed here extends that of Section 4.4.1, hence I refer to it as the *extended joint behavior consensus model* (eJCM). The first of three individual behavior models presented here is designed as an independent behavior version of the eJCM; thus, it is simply called the *extended individual behavior con-*

sensus model (eICM). The remaining two models are based on proposals and observations from the original experimental analysis (Kearns et al., 2009), and are labeled *proportional response model* (PRM) and *sticky proportional response model* (sPRM), respectively.

Extended Joint Behavior Consensus Model

In this section I formulate a potential function that captures the impact of past collective choices of i's conditioning set on its dependence neighborhood, its relative preference for each action, and its own past voting patterns. First, one can summarize a history $H^t_{\Gamma_i}$ of length h relevant to agent i, using the frequency function $f(a_i, H^t_{\Gamma_i})$ defined in Section 4.4.1 (with $\epsilon = 0.01$). Second, I employ the function $r_i(a_{N_i})$, defined in Section 4.4.1 as the product of $r_i(a_i)$ and a heuristic attenuation based on how many nodes in the dependence neighborhood currently vote differently. Observe that $r_i(a_{N_i})$ is increasing in the number of i's neighbors voting a_i, reflecting the positive influence of neighbor choices on i. A preliminary analysis of the human-subject data motivated an additional third factor, which captures agent i's own update history in an *inertia* term,

$$\mathcal{I}(a_i, H^t_i) = \begin{cases} t - \max_{\tau<t} \tau(1 - I(a^\tau_i, a_i)) & \text{if } a_i = a^{t-1}_i \\ [t - \max_{\tau<t} \tau(1 - I(a^\tau_i, a_i))]^{-1} & \text{otherwise} \end{cases}$$

In other words, one may model agent i's voting inertia as proportional to how long it has maintained its most recent action a^{t-1}_i. Inertia is treated as a factor tending to support retaining the same action. If the candidate action a_i is different from a^{t-1}_i, the term expresses the inverse of this inertia factor.

The potential function for agent i combines these terms,

$$\pi_i(a_{N_i} \mid H^t_{\Gamma_i}) = r_i(a_{N_i}) f(a_i, H^t_{\Gamma_i})^\gamma \mathcal{I}(a_i, H^t_i)^\beta, \tag{5.9}$$

where $\gamma, \beta \geq 0$ denote the weight or importance of the historical frequency $f(a_i, H^t_{\Gamma_i})$ and

the inertia $\mathcal{I}(a_i, H_i^t)$ relative to the estimated reward $r_i(a_{N_i})$. The normalized product of these potentials specifies joint behavior as described in (4.1). The model maintains two free parameters, β and γ.

Extended Individual Behavior Consensus Model

The eICM for consensus dynamics retains the main elements of eJCM (5.9), while imposing conditional independence among agents' actions given the common history. The result is a within-time dependence neighborhood N_i that contains only i itself for each i. The probabilistic eICM behavior is then given by:

$$\Pr(a_i \mid H_{\Gamma_i}^t) = \frac{1}{Z_i} r_i(a_i) f(a_i, H_{\Gamma_i}^t)^\gamma \mathcal{I}(a_i, H_i^t)^\beta.$$

The normalization ranges only over single-agent actions $a_i \in A_i$, thus Z_i is easy to compute for this model.

Proportional Response Model

Kearns et al. (2009) proposed the proportional response model, PRM, as a reasonably accurate predictor of their experiments' final outcomes. PRM specifies that agent i chooses action a_i at time t with probability proportional to $r_i(a_i) g(a_i, a_{\Gamma_i}^{t-1})$, where $g(a_i, a_{\Gamma_i}^{t-1})$ denotes the number of i's neighbors who chose a_i in the last time period,

$$\Pr(a_i^t \mid H_{\Gamma_i}^t) \propto r_i(a_i^t) g(a_i, a_{\Gamma_i}^{t-1}).$$

Sticky Proportional Response Model

Experiment data indicates a tendency among subjects to start with their preferred option, reconsidering their votes only after collecting additional information about their neighbors over several time periods. Therefore, I introduce the *sticky proportional response*

model, sPRM, which contains a parameter $\rho \in [-1, 1]$ reflecting an agent's stubbornness in maintaining its preferred option, regardless of observed neighbors' past choices. Intuitively, an agent's inherent bias toward its preferred option decays proportionally until there is no bias:

$$\Pr(a_i^t \mid H_{\Gamma_i}^t) \propto r_i(a_i^t) g(a_i, a_{\Gamma_i}^{t-1})(1 + \frac{I_{a_i}^{\max} \rho}{t}),$$

where $I_{a_i}^{\max} = 1$ if $a_i = \arg\max r_i(a)$ and $I_{a_i}^{\max} = 0$ otherwise.

5.2.2 Learning Graphical Structures

I employ the gradient ascent optimization technique outlined in Section 4.1 to learn the parameters of all model forms that maximize the likelihood of training data, given a fixed graphical structure. I next propose a greedy algorithm for learning both the graphical structure and parameters of an hGMM given observational data.

Each of the consensus voting experiments involves 36 human subjects. The largest dependence neighborhood size in these games ranges from 16 to 20, rendering computing exact data likelihood for a joint behavior model of this complexity (required for parameter learning described above) infeasible. Preliminary trials with the belief propagation approximation algorithm (Broadway et al., 2000) on these models, with $N = \Gamma = N^{\mathsf{O}}$, confirmed that the savings from approximation were insufficient for effective learning on this graph. Thus, one needs to employ models with simpler graphs in order to take advantage of hGMMs' expressiveness in representing joint behavior. Toward this end, I have developed a structure learning algorithm that produces graphs for hGMMs within specified complexity constraints.

Though dictated by computational necessity, automated structure learning has additional advantages. As noted above, the observation graph N^{O} may not constitute the ideal structure N for a predictive graphical model for agent behavior. Since actual agent behavior is naturally conditioned on its observable history, assume that the *conditioning set* coincides with the observation dependence neighborhood, $\Gamma = N^{\mathsf{O}}$. Nevertheless, once

the history representation is abstracted it may well turn out that non-local historical activity provides more useful predictive information. If so, the structure of the learned graph that defines each i's within-time dependence neighborhood may provide interesting insights on the agents' networked behavior.

The structure learning algorithm detailed below addresses the problem of learning N_i for every i, taking $\Gamma_i = N_i^{\mathsf{O}}$ as fixed. Note that the individual behavior multiagent models described in Section 4.2 impose $N_i = \{i\}$ for each i, and thus do not need to learn the within-time graphs. Starting from an empty graph, the algorithm greedily adds edges to improve the log-likelihood $L_{\mathsf{hG}}(X;\theta)$ of the training data X, subject to a constraint that the maximum node degree not exceed a specified bound $d_{\max}$. Since the set of edges E is the only structural model feature that changes during the local search process, the algorithm uses $L_E(X;\theta)$ to abbreviate $L_{\mathsf{hG}}(X;\theta)$ (defined in (4.2)), as induced by the hGMM $\mathsf{hG} = (V, E, A, \Gamma, \pi)$. I have found that the optimal settings of these parameters $\theta = (\beta, \gamma)$ is insensitive to within-time dependencies, hence I apply the parameter learning operation only once, at the beginning of the search process. The algorithm is defined formally below.

Algorithm 1 Learn hGMM

Input $X, (V, A, \Gamma), d_{\max}$
$E \leftarrow \emptyset$
Use gradient descent to identify $\theta \approx \arg\max L_E(X;\theta)$.
$\tilde{E} \leftarrow \{(i,j) \mid i \in V, j \in V\}$
repeat
  $newedge \leftarrow$ **false**
  $(i_*, j_*) \leftarrow \arg\max_{(i,j)\in\tilde{E}} L_{E\cup(i,j)}(X;\theta)$
  if $L_{E\cup(i_*,j_*)}(X;\theta) \geq L_E(X;\theta)$ **then**
    $E \leftarrow E \cup \{(i_*, j_*)\}$
    $newedge \leftarrow$ **true**
  end if
  $\tilde{E} \leftarrow \tilde{E} \setminus \{(i_*, j_*)\} \setminus \{(i,j) \mid \max(|N_i|, |N_j| = d_{\max}\}$
  Use gradient descent to identify $\theta \approx \arg\max L_E(X;\theta)$.
until $\tilde{E} = \emptyset \vee newedge =$ **false**

I evaluate the learned multiagent models by their ability to predict future outcomes, as

represented by a test set Y. Given two models M_1 and M_2, one can compute their corresponding log-likelihood measures for the test data set Y: $L_{M_1}(Y)$ and $L_{M_2}(Y)$. Note that since log-likelihood is negative, I instead examine the negative log-likelihood measures, which means that M_1 is better than M_2 predicting Y if $-L_{M_1}(Y) < -L_{M_2}(Y)$, and vice versa.

5.2.3 Experiment

I empirically evaluate the predictive power of eJCMs in comparison with eICMs, PRMs, and sPRMs, using the consensus dynamics experiment data from Kearns et al. (2009). I also examine the graphs induced by structure learning, and relate them to the corresponding observation networks by various statistical measures.

Experiment Settings

The human-subject experiments are divided into nine different sets, each associated with an observation network structure. These networks differ qualitatively in various ways, characterized by node degree distribution, ratio of inter-group and intra-group edges, and the existence of a well-connected minority (Kearns et al., 2009). In particular, networks whose edges are generated by an Erdos-Renyi (ER) (Erdos and Renyi, 1959) process have a notably more heavy-tailed degree distribution than those generated by a preferential attachment (PA) process (Barabási and Albert, 1999). For each trial, subjects were randomly assigned to nodes in the designated network structure, and preferences based on one of three possible incentive schemes. Since subjects in these experiments can change their votes at any time, the resulting data is a stream of asynchronous vote actions. I discretize these streams for data analysis, recording the subjects' votes at the end of each time interval of length δ seconds. The experiments examine interval lengths $\delta \in \{0.5, 1.5\}$.

I learn predictive models for each experiment network structure, pooling data across subject assignments and incentive schemes. My premise is that network structure is the

main factor governing the system's collective behavior, in line with the findings of Kearns et al. (2009). In each experiment set, I use eight of the nine trials for training the predictive models for each form. The within-time graphs are learned with node degree constraint $d_{\max} = 10$. I evaluate the models based on their predictions over a test set comprising the left-out experimental trial. This process is repeated five times, with a different randomly chosen trial reserved for testing. Each data point in the reported empirical results averages over these five repetitions.

Using the labels of Kearns et al. (2009), I distinguish three experiment networks according to their graph generation processes and the existence of a minority group of well-connected nodes that share the same vote preference (see Table 5.1).

Table 5.1: Voting Experiment Settings

Label	Strong Minority	Graph Generator
coER_2	No	Erdos-Renyi
coPA_2	No	Preferential attachment
power22	Yes	Preferential attachment

Predictions

I first examine predictions of subjects' votes across time, conditional on available history. Figure 5.7 presents predictive performance for eJCMs, eICMs, PRMs, and sPRMs, measured by negative log-likelihood of the test data according to the respective models. eJCMs perform best in predicting dynamic behavior in the consensus dynamics experiments for all three experiment settings, given data discretized at interval lengths of 0.5 and 1.5 (differences significant at $p < 0.025$). Both the eJCM and eICM representations, which share similar fundamental elements, handily outperform PRM and its sticky version sPRM.

Contrary to my expectation, eJCMs and eICMs that employ only the last period of

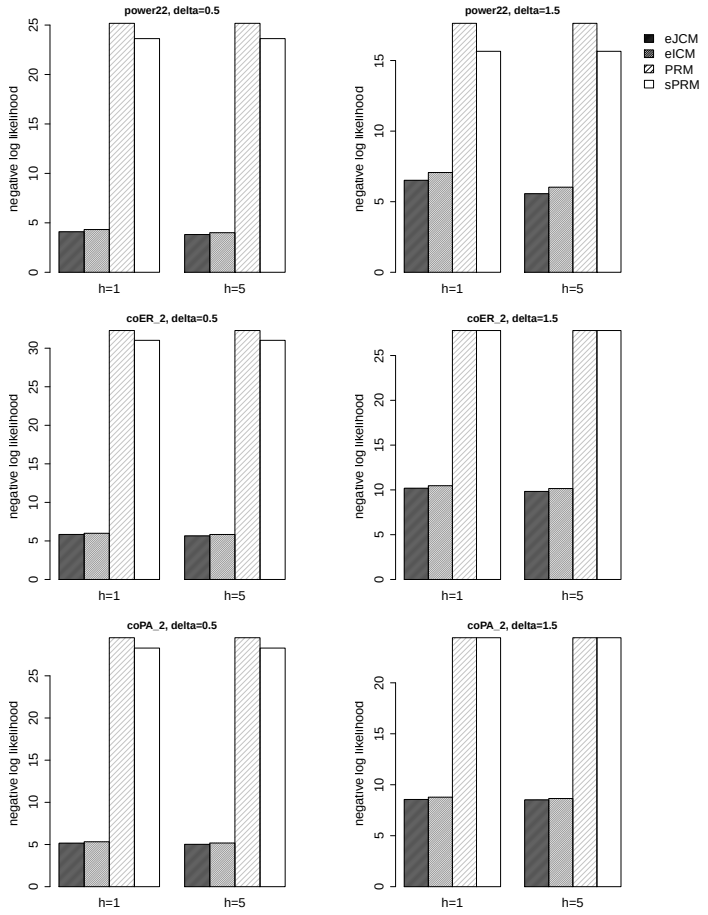

Figure 5.7: eJCMs provide better predictions of the system's dynamics than eICMs, PRMs, and sPRMs in twelve settings: the three experiment networks power22 (top), coER_2 (middle), and coPA_2 (bottom), each for two history lengths, using time discretization intervals $\delta = 0.5$ (left) and $\delta = 1.5$ (right). The prediction quality differences between eJCM and eICM are significant ($p < 0.025$) in all scenarios.

historical data ($h = 1$) predict as well as those with $h = 5$. This is probably attributable in part to the heuristic nature of the frequency function (4.4), but may also indicate a tendency among subjects to account for only the most recent state of neighbors when choosing their own actions. Unsurprisingly, all models perform worse with the larger temporal aggregation interval $\delta = 1.5$. More salient is that the relative performance across models is qualitatively identical for the two δ settings, which lends support to the robustness of my findings. The results in general demonstrate eJCMs' ability to capture joint dynamic behavior, especially behavior interdependencies induced by limited historical information.

I next evaluate the models' ability to predict the end state of a consensus dynamics run: whether consensus is reached, and if so which one. For a particular model M, I start a simulation run with agents choosing their preferred colors, and then draw joint behaviors from M for each time period until a consensus is reached or the time limit is exceeded. I average over 100 run instances for each environment setting and model. As no considerable differences were observed in the models' end-game predictions for different history lengths, I display results for only $h = 1$ henceforth.

The proportion of simulation instances reaching consensus under the eICM and PRM models corresponds well with observed experiment results, as shown in Figure 5.8.[7] Simulated runs drawn from eJCMs converge to consensus at markedly lower rates in general. The eJCM end-game predictions improve with greater $\delta = 1.5$, especially in the power22 setting where they match the experiment outcomes almost exactly. A closer look at the end-game prediction results reveals that the individual behavior models' predictions on the final consensus color are considerably out of line with the actual experiments for both coER_2 and power22. eJCMs, on the other hand, provide dramatically more accurate predictions on the consensus color in the power22 setting. The ratio between blue and red consensus instances by eJCMs in coPA_2 resembles that of the actual experiments more than eICMs and PRMs. In the coER_2 setting all models' predictions on the favored consensus color (blue) miss the actual experiments' favored consensus color (red), though the ratio of red-to-blue consensus predicted by eJCM is less skewed than that of eICMs and PRMs.

Last, I demonstrate the benefits of decoupling dependence neighborhood and conditioning set by comparing eJCM against oJCM, which also learns the within-time dependence neighborhood from data, but does not differentiate Γ and N by assuming that $\Gamma = N$. Figure 5.9 shows that oJCMs perform worse than both eJCMs and eICMs in predicting the system's votes across time as well as end-game results, for the power22 setting with $h = 1$ and $\delta = 0.5$.[8] Moreover, note that the graphs learned by oJCMs contain disconnected node

[7] End-game results from sPRMs are similar to those from PRMs, and not shown here.

[8] I obtain similar results for oJCMs in other experiment settings and environment parameters, not shown

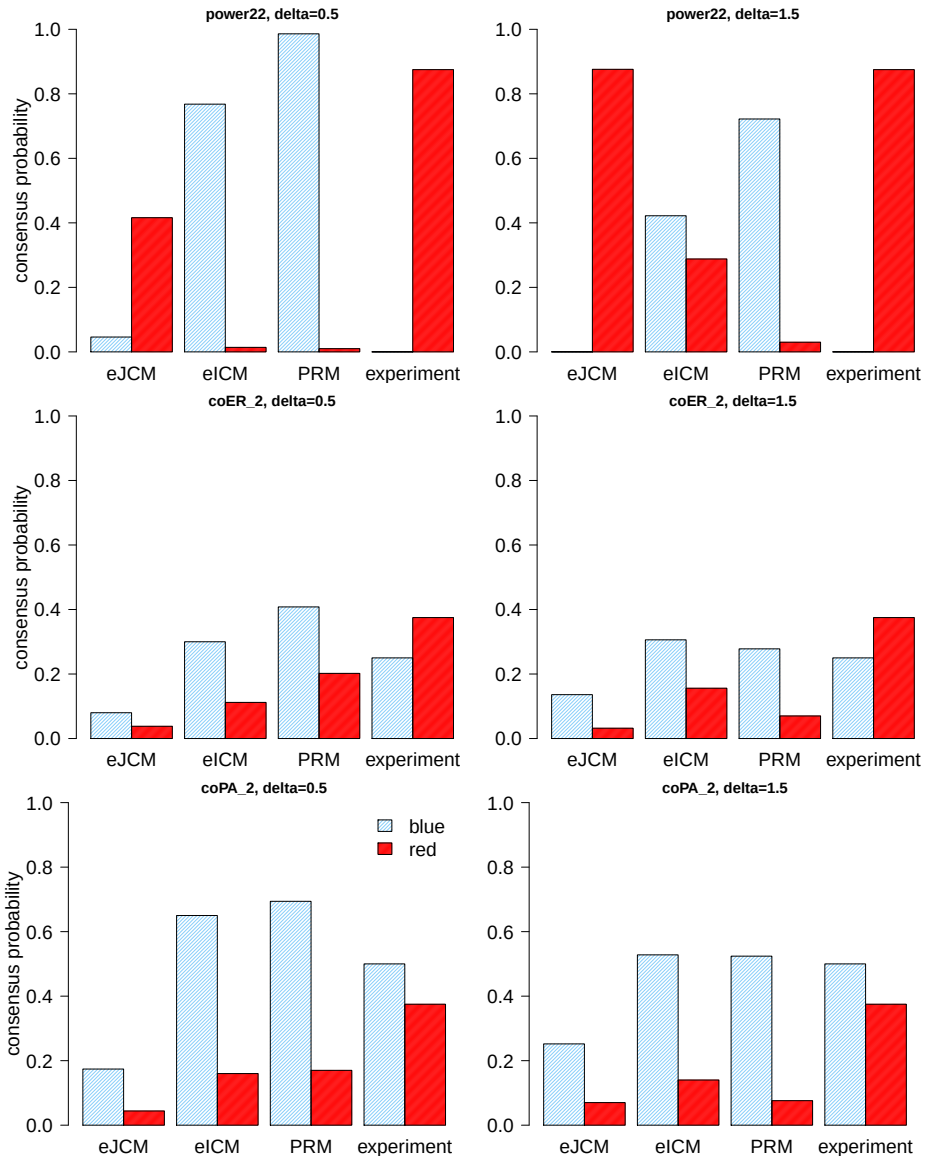

Figure 5.8: eJCM predictions on the probability of reaching consensus are lower than predictions from eICMs and PRMs, as well as experiment outcomes, with different time discretization intervals $\delta = 0.5$ (left) and $\delta = 1.5$ (right). However, the eJCM is significantly more accurate than eICMs or PRMs on predicting the ultimate consensus colors.

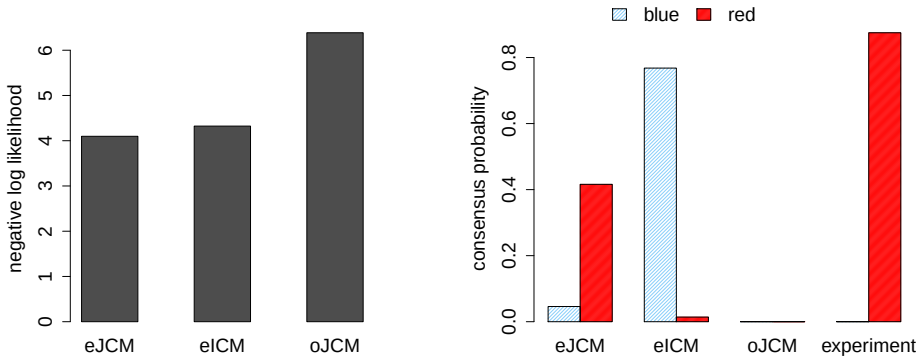

Figure 5.9: oJCMs provide worse predictions than eJCMs and eICMs of both system dynamics and end-game results (power22, $h = 1$, $\delta = 0.5$).

subsets, which potentially prevent vote decisions from propagating throughout the network, causing failures in producing any consensus instances.

Graph Analysis

This section provides a characterization the learned edges that define N in the eJCM representation, and discover connections between the learned graphs and the aforementioned prediction results. First, one can categorize edges by their end-point nodes' vote preferences: I refer to edges that connect two red-favoring (blue-favoring) nodes as *intra* red-favoring (blue-favoring), and those between red-favoring and blue-favoring nodes as *inter* edges. Figure 5.10 presents the proportion of each edge type in both the given observation graphs and the learned within-time graphs. Whereas a majority of edges in the observation graphs are inter edges, the within-time graphs that define N consist mostly of intra edges. That is, there are more interdependencies in eJCMs among agents of the same preference than among conflicting agents. The ability to discover these inter edges and incorporate the information they carry in its joint action distribution may help the eJCM representation to better capture dynamic behavior and end-game results, as illustrated and discussed in Section 5.2.3. For the power22 setting in particular, eJCMs often assign a majority of edges as intra red, and thus effectively identify the presence of a strongly connected

here.

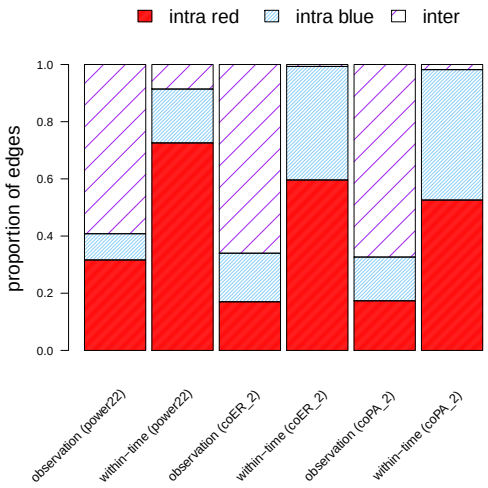

Figure 5.10: Distributions of edges from three different categories, intra red, intra blue, and inter, in the given observation and learned within-time graphs for eJCM ($\delta = 0.5$).

red minority who dictated end-game colors in the actual experiments. This construction allows eJCMs to predict end-game consensus colors much more accurately than eICMs and PRMs, which rely entirely on the observation graphs.

I further investigate whether these proportion measures provide any predictions on the number of consensus instances induced by eJCMs. I pool data from all experiment settings—power22, coPA_2, and coER_2—and compute a simple linear regression of the number of red (blue) consensus instances with respect to the proportion of intra red (blue) edges. The resulting regression coefficients are statistically significant for both blue and red ($p < 0.05$). Figure 5.11 suggests that a weak positive correlation between the within-time graphs' intra edges and the number of consensus instances. Intuitively, more interdependence between same-preference nodes allows them to have more influence on one another, helping to diffuse vote choices more rapidly throughout the system.

I next examine eJCM edges in terms of how far apart are their end-nodes in the observation graph. Let $\phi_{i,j} \geq 1$ denote the length of shortest path from i to j in the observation graph. Given a graph G on the same set of nodes, one can calculate the proportion of edges in G that connect nodes separated by a certain distance in the original observation graph.

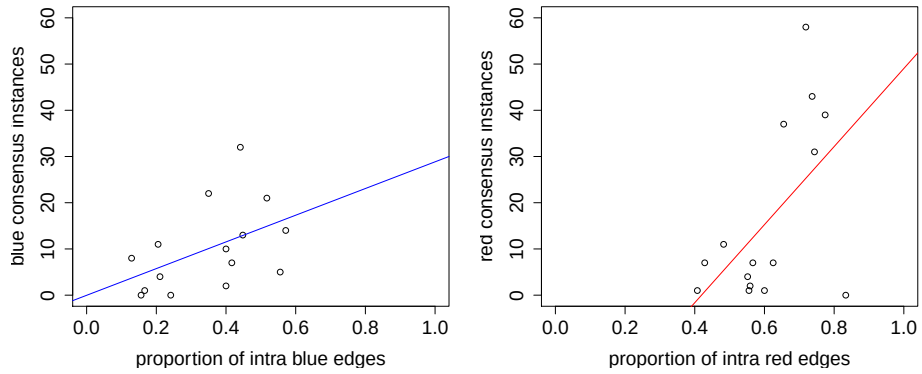

Figure 5.11: The number of consensus instances in blue (left) and red (right), and proportion of eJCM intra edges of the corresponding colors.

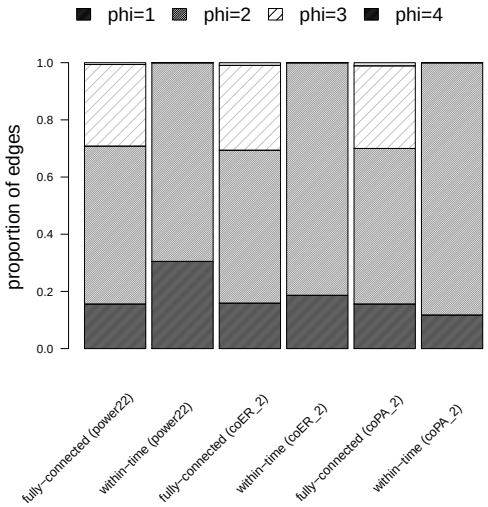

Figure 5.12: Distributions of edges in the within-time graphs based on the distance between their end-nodes ϕ in the observation graph ($\delta = 0.5$).

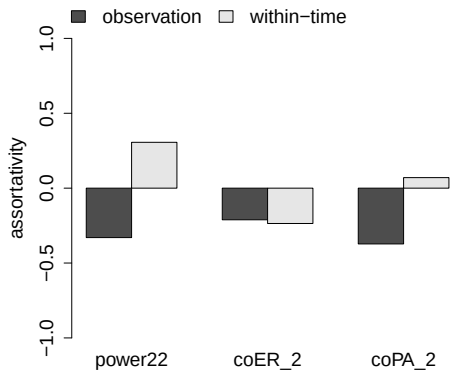

Figure 5.13: Assortativity of the observation graphs and the learned within-time graphs ($\delta = 0.5$).

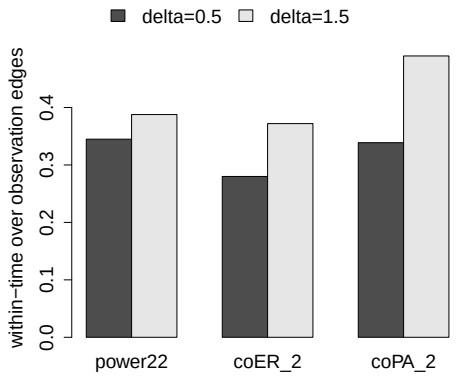

Figure 5.14: Sparsity of the within-time graphs.

Figure 5.12 presents the profile of such distances for pairs of nodes in the learned eJCMs. For comparison, the profiles labeled "fully connected" simply reflect the distribution of node distances in the original observation graph: most of the nodes are one hop apart or closer ($\phi \leq 2$), with a modal distance $\phi = 2$. A large majority of edges in the learned within-time graphs have $\phi = 2$, that is, are close but not directly linked in the observation graphs.

Next I compare the *assortativity* (Newman, 2003) of the learned and original graphs. A graph G's assortativity coefficient in $[-1, 1]$ captures the tendency for nodes to attach to others that are similar (positive values) or different (negative values) in connectivity. As illustrated in Figure 5.13, the large difference in assortativity for the power22 setting stresses eJCMs' ability to discover interdependencies among agents' actions that are not

captured in the observation graph. More specifically, the resulting eJCMs are able to capture correlations in actions among nodes of similar degrees in the power22 setting, where the minority nodes are more densely connected than the majority, confirming the findings by aforementioned graph analyses on intra and inter edges. I also investigate the sparsity of the learned graphs for different values of δ. The sparsity measure used here is the number of edges in the learned within-time graph divided by the number of edges in the corresponding observation graph. Figure 5.14 illustrates that the within-time graphs become sparser as the discretization interval shrinks from 1.5 to 0.5 in all experiment settings. Intuitively, the finer grain the discretization is, the fewer simultaneous vote changes there are in one time period. As a result, there may be fewer interdependencies among agents' actions.

One of the main results here is a demonstration of the feasibility of learning probabilistic models of dynamic multiagent behavior from real traces of agent activity on a network. Another accomplishment is the introduction and evaluation of a greedy structure-learning algorithm to induce within-time dependencies from time-series data, which effectively serves as a tool for constructing this demonstration. This study's investigation finds that the learned joint behavior model provides better predictions of dynamic behavior than several individual behavior models, including the proportional-response models suggested in the original experimental analysis. This provides evidence that expressing joint behavior is important for dynamic modeling, even given partial history information for conditioning individual behavior. The graph analysis further reveals characteristics of the learned within-time graphs that provide insights about patterns of agent interdependence, and their relation to structure of the agent interaction network.

CHAPTER VI

Application: Information Diffusion in Networks with Unobserved Links

The empirical studies of history-dependent graphical multiagent models presented in previous chapters have exclusively focused on capturing interdependence emerging from history abstraction. In this chapter, I discuss hGMMs' capabilities to model another type of agent interdependence caused by the modeler's lack of information about the graphical structures that capture agent interactions. I particularly focus on the problem of modeling *information diffusion* on networks with missing or unobserved edges. Note that unlike previously studied multiagent modeling problems in this thesis, there is no function of agent utility explicitly defined in this problem domain.

Information diffusion can be seen at work in a wide range of scenarios, such as news propagation, political grass-root campaigns, product promotion, and technology adoption. Models of information diffusion on social networks, for instance, can inform viral marketing campaigns by offering predictions and analyses of product popularity and advertising effectiveness, given information about early buyers' behaviors (Leskovec et al., 2007a). They can also be employed to assess more abstract properties, such as the likelihood of information diffusing from one node to another node separated by a given distance (Song et al., 2007; Choudhury et al., 2008; Bakshy et al., 2009), or the conditions for information survival in a dynamic network (Chakrabarti et al., 2007).

Applying such models requires knowledge of the network structure, typically inferred based on data on affiliation, communication, or other observable traits of agent relationships. In practice, however, such evidence is generally incomplete and uncertain. Sophisticated learning methods may improve detection in such circumstances (Adar and Adamic, 2005; De Choudhury et al., 2010; Backstrom and Leskovec, 2011), but inevitably, real-world agents will have connections that are unobserved by specific third parties. For example, Internet social networks capture only a portion of all human interactions. Thus, any model of information diffusion must account for propagation of information along paths not included in the observed network. This study addresses the problem of modeling information diffusion when the network is only partially observed, and investigates two approaches. The first learns graphical model potentials for a given network structure, compensating for missing edges through induced correlations among node states. The second attempts to learn the missing connections directly.

6.1 Information Cascade

In the now standard approach to modeling information diffusion on networks (Kleinberg, 2007), a node or agent is in one of a finite set of states (e.g., having particular information and adopting a technology) at a given time. In each discrete time period, each agent decides what state to adopt in the next period, as a probabilistic function of the states of its neighbors and itself. Formally, let G be a network of n agents over a discrete time horizon T. Let us restrict attention here to capturing the diffusion of a single bit of information, so at time t, each agent i can be in either of two states: $a_i^t = 1$ indicates that agent i has been *infected*, which entails receiving and retaining the spreading information, and $a_i^t = -1$ otherwise. An undirected edge (i, j) in G represents a connection between agents i and j: conceptually, that i and j interact and may be aware of each other's state. For each node i, J_i denotes the *interaction neighborhood* of i: $J_i = \{j \mid (i, j) \in G\} \cup \{i\}$. Let $a^t = a^t_{\{1,\ldots,n\}}$ be the state of all agents at time t. Note that I overload a, which denotes

action in the multiagent system framework throughout previous chapters, to represent *state* in the diffusion scenario, since both states and actions of an agent can be simply mapped to *values* of graphical model *variables*.

In the version of information diffusion studied here, infection *persists*: there exists no agent i and time t such that $a_i^t = 1$ and $a_i^{t+1} = -1$.[1] I define a binary feature $b(a_i^t, a_i^{t-1}) = 1$ to indicate that i becomes infected at time t ($a_i^{t-1} = -1$ and $a_i^t = 1$), otherwise $b(a_i^t, a_i^{t-1}) = -1$. I also define t_i^* as the time when i becomes infected, and $c(a_{J_i}^t) = |\{j \in J_i \mid a_j^t = 1\}|$, the count of agents in J_i who are infected at t. Similarly, let $\sigma(a_{J_i}^t, a_{J_i}^{t-1}) = |\{j \in J_i \mid b(a_j^t, a_j^{t-1}) = 1\}|$, the number of agents in J_i that *become* infected at time t.

The most popular model of infection is the *cascade model* (Kempe et al., 2003), in which the tendency to infect increases with the proportion of infected neighbors. Goldenberg et al. (2001) proposed a cascade model form that specifies the probability of infection as:

$$\begin{aligned} \chi(a_i^t = 1 \mid a_i^{t-1} = 1, a_{J_i - \{i\}}^{t-1}) &= 1 \\ \chi(a_i^t = 1 \mid a_i^{t-1} = -1, a_{J_i - \{i\}}^{t-1}) &= 1 - (1-\alpha)(1-\beta)^{c(a_{J_i}^{t-1})}. \end{aligned} \tag{6.1}$$

In this model, $\beta \in [0, 1]$ represents node i's tendency of infection from interacting with one of its infected neighbors, and $\alpha \in [0, 1]$ reflects the possibility of node i getting information from sources other than its neighbors, or in other words *spontaneously* becoming infected.[2] For example, when $\alpha = 0$, $\beta > 0$, and $c(a_{J_i}^{t-1}) = 1$, i does not get information from sources other than its neighbors, and can become infected only from interacting with its sole infected neighbor with probability $1 - (1 - \beta) = \beta$.

[1] The persistence and symmetry properties of infection in this study are standard in network science literature.

[2] The cascade model entails that influence is uniform across a node's neighbors and all network nodes will eventually become infected. I adopt this model not because these characteristics are necessarily desired, but because the model is widely applied in the literature. Nothing about my approach is particularly geared to these properties.

Stonedahl et al. (2010) introduced an alternative cascade model, which is labeled C, that induces similar behavior based on the same intuitions, but with a simplified probability expression:

$$\chi(a_i^t = 1 \mid a_{J_i}^{t-1}) = \alpha + \beta \frac{c(a_{J_i}^{t-1})}{|J_i|}. \tag{6.2}$$

Assuming that agent states are conditionally independent given past states, the cascade models define a joint distribution of states at time t:

$$\Pr(a^t \mid a^{t-1}) = \prod_i \chi(a_i^t \mid a_{J_i}^{t-1}). \tag{6.3}$$

Although the cascade model allows for positive probability of infection even if no known neighbors are infected, its accuracy suffers if the network structure is missing connections. Gomez-Rodriguez et al. (2010) were first to identify and tackle the problem of discovering underlying network structure given only diffusion history. They introduce an algorithm called `NetInf`, which identifies a network that optimizes an approximate measure of fit to observed infection times.[3] This research work examines the *structure learning* approach: a greedy algorithm of my own, named `MaxLInf`, as well as a version `NetInf′` of the existing algorithm `NetInf`, modified to fit the problem setup introduced here. I further introduce a fundamentally different approach, *potential learning*, which can compensate for missing edges by capturing induced correlations of behavior in potential functions that encompass nodes not directly connected in the given graph.

6.2 Problem Definition

Let G be the observed network, and $S = \{s\}$ a set of diffusion instances or traces, each of length T, $s = \{a^0, \ldots, a^T\}$. Assume that the diffusion was generated by some

[3] Gomez-Rodriguez et al. (2011) introduced an improved version of `NetInf` that treats the underlying diffusion as a continuous-time process and learns the model parameters from data.

process (in the following empirical study, the cascade model) on a true underlying network $G^* \supseteq G$. Given G and S, one may want to make inferences about future diffusion events. I next provide an example of the problem and formally define the study's objectives.

6.2.1 Example

Consider the example four-node scenario shown in Figure 6.1, where the edge between nodes 1 and 3 is not observed. For simplicity, let us focus on the setting of zero spontaneous infections: nodes can become infected only from interacting with their neighbors. In the example run, the spread started with node 1 at time $t = 1$ and reached nodes 4 and 3 at $t = 5$ and $t = 6$, respectively. The missing edge $(1, 3)$ may mislead one to interpret node 3's infection as spontaneous. Section 6.3 provides more details about how each approach in this study compensates for the missing connection $(1, 3)$ in this particular example.

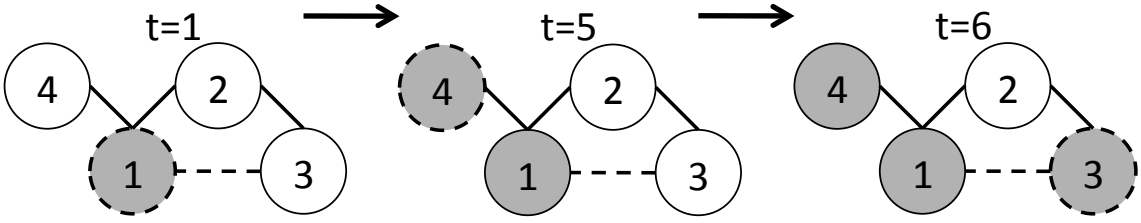

Figure 6.1: A four-node scenario with one missing edge (dashed). Infected nodes are shaded. Newly infected nodes at each time period are marked with dashed rims.

6.2.2 Evaluation

One can capture the dynamics of information diffusion using the joint probability distribution of agent states, conditional on past states. Notationally, $\Pr^M(a^t \mid a^{t-1})$ is the joint distribution of agent states at time t induced by a particular model M. Given a set S of m

traces, one can compute the data's average logarithmic likelihood as:

$$L^M(S) = \frac{1}{m} \sum_{s \in S} \frac{1}{T} \sum_t \log \Pr{}^M(a^t \mid a^{t-1}). \tag{6.4}$$

As in many information network studies, I also focus on predicting the aggregate extent of future network infection, given some initial diffusion data. In particular, given an initial network state a^0, one can sample from the model M diffusion traces of length $\hat{T}$ that all start with a^0. These traces define an empirical distribution over the fraction $c(a^{\hat{T}})/n$ of nodes that are infected at time $\hat{T}$, denoted as $Q^M(c(a^{\hat{T}})/n \mid a^0)$. I employ the skew divergence (Lee, 1999), a version of the Kullback-Leibler (KL) divergence metric, to measure the distance between the two discrete value distributions induced by the true model M^* and by the test model M, conditional on observations of the initial state a^0. The KL divergence is specified as:

$$d_{KL}(Q^{M^*}, Q^M \mid a^0) = \sum_{c=0}^{n} Q^{M^*}(c/n \mid a^0) \log \frac{Q^{M^*}(c/n \mid a^0)}{Q^M(c/n \mid a^0)},$$

and average skew divergence given data S,

$$D(Q^{M^*}, Q^M \mid S) = \frac{1}{m} \sum_{a^0 | s \in S} d_{KL}(Q^M, \rho Q^{M^*} + (1-\rho) Q^M \mid a^0), \tag{6.5}$$

with ρ set at 0.99 to avoid the problem of undefined KL divergence values (Lee, 1999).

One can further measure the quality of the graphs learned in simulated scenarios where access to the true underlying graph is available. In particular, I employ the following metrics on structural differences between the true underlying graph $G^* = (V, E^*)$ and the learned graph $G' = (V, E')$:[4]

$$\delta_-(G^*, G') = \frac{|E^* \setminus E'|}{|E^*|}, \text{ and } \delta_+(G^*, G') = \frac{|E' \setminus E^*|}{|E'|}.$$

[4]These metrics were also employed in evaluating the original `NetInf` (Gomez-Rodriguez et al., 2010).

Here $\delta_-(G^*, G')$ represents the proportion of false negative edges with respect to the true graph G^*, or in other words, edges that remain missing in the learned graph G'. Similarly, $\delta_+(G^*, G')$ captures the proportion of false positive edges in the learned graph G'.

6.3 Dealing with Partially Observed Networks

Consider two broad approaches to learning diffusion models despite unobserved network edges. The potential learning approach captures probabilistic dependencies induced by the missing edges in the potential function of a graphical model. The structure learning approach attempts to recover the edges explicitly by learning network structure.

6.3.1 History-Dependent Graphical Multiagent Models

Unlike the cascade models, history-dependent GMMs, as described in Chapter IV, do not assume conditional independence of agent states given history, but specify the joint state directly. That hGMMs compute the potentials of all state configurations of a neighborhood allows reasoning about state correlations between neighbors, which appears to compensate for missing edges.

Figure 6.2 illustrates an example four-agent hGMM of the scenario described in Section 6.2.1. Note that the figure distinguishes two kinds of edges. As defined in Section 4.1, undirected edges define the neighborhoods N_i within a time slice, capturing correlations among the nodes in N_i at a particular time t. Directed edges ending at i capture how past states of the conditioning set Γ_i influence i's present state. The resulting potential function, $\pi_i(a^t_{N_i} \mid a^{t-1}_{\Gamma_i})$, allows that the conditioning nodes be different from the nodes interdependent within a time slice. In contrast, a cascade model of the same scenario would omit the undirected edges, expressing the cascade functional form $\chi(a^t_i \mid a^{t-1}_{\Gamma_i})$. For most of this chapter, I similarly assume that, without loss of generality, N_i, Γ_i, and the given J_i are the same for all i, refer to them all as N_i, and employ the potential function form $\pi_i(a^t_{N_i} \mid a^{t-1}_{N_i})$.

Given data from the example scenario of Section 6.2.1, one would learn a high value for

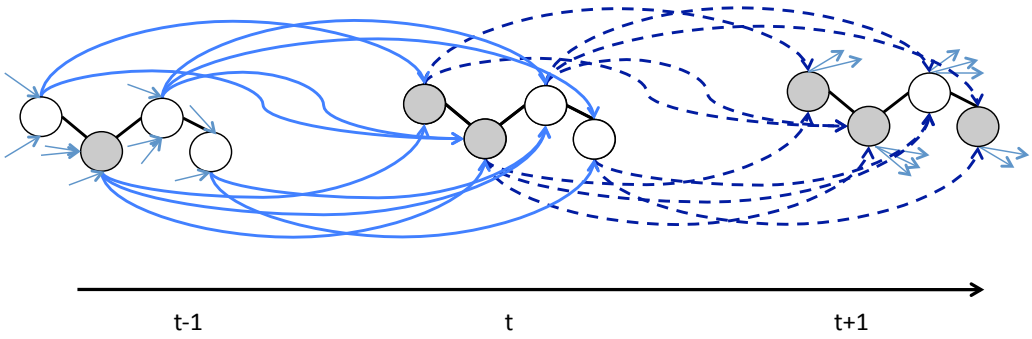

Figure 6.2: A history-dependent graphical multiagent model of four nodes. Undirected edges capture correlations among nodes of the same time point. Directed edges capture the conditioning of each node's state on its conditioning set's previous states: dashed arrows connect nodes from t to $t+1$, and solid arrows link nodes from $t-1$ to t.

α for the cascade models, assuming spontaneous infections, even though in the true model, information spreads only from node to node. In contrast, hGMMs could use the potential function of node 2 to express correlations between nodes 1 and 3 to compensate for the missing edge $(1, 3)$. For instance, consider neighborhood $N_2 = (1, 2, 3)$ with a short one-time-period history $a^5_{N_2} = (a^5_1, a^5_2, a^5_3) = (1, -1, -1)$, as depicted in Figure 6.1. Since the potentials $\pi(a^6_{N_2} = (1, -1, -1) \mid a^5_{N_2})$ and $\pi(a^6_{N_2} = (1, -1, 1) \mid a^5_{N_2})$ govern how node 1 infects node 3, assigning positive values to these potentials enables hGMMs to capture the interaction. Unlike $\pi(a^6_{N_2} \mid a^5_{N_2})$, the corresponding cascade-model construct $\chi(a^6_2 \mid a^5_{N_2})$ cannot compensate in this way because it inherently assumes independence among nodes in N_2.

I introduce hGMMs for information diffusion in two different forms.

6.3.1.1 Tabular hGMMs

The *tabular hGMM* $\mathbf{tabG}$ explicitly specifies the potential function of each neighborhood, $\pi(a^t_{N_i} \mid a^{t-1}_{N_i})$, as a function of five features:

$$c(a^{t-1}_{N_i}),\ \sigma(a^t_{N_i},\ a^{t-1}_{N_i}),\ |N_i|,\ a^{t-1}_i,\ a^t_i.$$

For learning such forms, the potential value corresponding to each configuration of these features is treated as a parameter. The number of parameters thus grows polynomially with the largest neighborhood's size.

6.3.1.2 Parametric hGMMs

The *parametric hGMM* **paG** employs a functional form for potentials based on the cascade model formula (6.2). I define $g(a_y = 1 \mid a_Y)$ as a probability measure of any uninfected node y in the set of nodes Y becoming infected given the current state a_Y: $g(a_y = 1 \mid a_Y) = \alpha + \beta c(a_Y)/|Y|$. I overload $g(a_Y) = g(a_y = 1 \mid a_Y)$, as g is identical for all y. Let $N_{-i} = N_i \setminus \{i\}$ and $c = \sigma(a^t_{N_{-i}}, a^{t-1}_{N_{-i}})$. The potential function of neighborhood N_i is the product of three terms:

$$\pi(a^t_{N_i} \mid a^{t-1}_{N_i}) = \chi(a^t_i \mid a^{t-1}_{N_i}; \alpha, \beta) g(a^{t-1}_{N_{-i}}; \alpha_1, \beta_1)^{|c-\gamma|N_i||} \left(1 - g(a^{t-1}_{N_{-i}}; \alpha_{-1}, \beta_{-1})\right)^{|N_{-i}|-c}. \quad (6.6)$$

The first term of (6.6) represents the probability of node i's infection, given by the cascade model (6.2). The second term corresponds to nodes in N_{-i} that become infected at t, whereas the final term corresponds to nodes in N_{-i} not infected at t. If one assumed independence of agent states, the exponent in the second term would simply be c, as the joint probability of c nodes from Y becoming infected is $g(a_Y)^c$. A preliminary study of **tabG** suggests that the distance of this count from a fraction of neighborhood size provides a good way to capture correlation among neighbor infections, hence the exponent in the second term $|c - \gamma|N_i||$ with $\gamma \in [0, \infty)$. Here γ indicates the degree of correlation among agent states. Thus, complete independence of agent states results in $\gamma = 0$. Overall, the **paG** model employs seven parameters: α, β, α_1, β_1, α_{-1}, β_{-1}, and γ.

The graphical multiagent model approach does not explicitly seek to discover missing edges, but instead takes advantage of the GMM's flexibility in capturing joint states to

model information diffusion on the original network.

6.3.2 The Ability of Joint Behavior Multiagent Models to Compensate for Missing Edges: An Example

Here I demonstrate how expressing joint states enables hGMMs to compensate for missing edges. In particular, I examine the theoretical limitations of a parametric hGMM and a cascade model designed for the example scenario described in Section 6.2.1 and illustrated in Figure 6.1. For simplicity, I solely focus on the subnetwork of nodes 1, 2 and 3, given that node 1 is already infected at time $t = 1$, and on predicting diffusion observations at time $t = 2$. As in the aforementioned example in Section 6.2.1, the underlying cascade process is presumably defined by $(\alpha = 0, \beta > 0)$ as described in the cascade function (6.2). The probability of each network's state under this generative model at $t = 2$ is detailed in Table 6.1

(a_2^2, a_3^2)	Probability
$(-1, -1)$	$[(1 - \frac{\beta}{3})][(1 - \frac{\beta}{3})]$
$(1, -1)$	$[\frac{\beta}{3}][(1 - \frac{\beta}{3})]$
$(-1, 1)$	$[(1 - \frac{\beta}{3})][\frac{\beta}{3}]$
$(1, 1)$	$[(\frac{\beta}{3})][\frac{\beta}{3}]$

Table 6.1: Probability of nodes 2 and 3's state configurations, specified by the underlying cascade model ($\alpha = 0, \beta > 0$). Terms corresponding to node 2 are grouped in the first square brackets. Those in the second square brackets originate from node 3.

Let K be a cascade model based on the network structure without edge $(1, 3)$. Its diffusion parameters are denoted as $(\alpha_{\mathsf{K}}, \beta_{\mathsf{K}})$. Similarly, I define H as a parametric hGMM based on the same network structure, whose parameters are $(\alpha_{\mathsf{H}}, \beta_{\mathsf{H}}, \alpha_{1,\mathsf{H}}, \beta_{1,\mathsf{H}}, \alpha_{-1,\mathsf{H}}, \beta_{-1,\mathsf{H}}, \gamma_{\mathsf{H}})$ respectively. I further assume that $\alpha_{\mathsf{H}} = \alpha_{1,\mathsf{H}} = \alpha_{-1,\mathsf{H}}$ and $\beta_{\mathsf{H}} = \beta_{1,\mathsf{H}} = \beta_{-1,\mathsf{H}}$, effectively allowing H to be identified by the set of only three parameters $(\alpha_{\mathsf{H}}, \beta_{\mathsf{H}}, \gamma_{\mathsf{H}})$. Table 6.2 displays the probability distributions of the network's joint state at $t = 2$, as specified by the

cascade function (6.2) and the parametric hGMM potential function (6.6).

(a_2^2, a_3^2)	Probability by K	(Unnormalized) Probability by H
$(-1,-1)$	$[1-\alpha_{\mathsf{K}}-\frac{\beta_{\mathsf{K}}}{3}][1-\alpha_{\mathsf{K}}]$	$[(1-\alpha_{\mathsf{H}}-\frac{\beta_{\mathsf{H}}}{3})(1-\alpha_{\mathsf{H}}-\frac{\beta_{\mathsf{H}}}{2})][(1-\alpha_{\mathsf{H}})(1-\alpha_{\mathsf{H}})]$
$(1,-1)$	$[\alpha_{\mathsf{K}}+\frac{\beta_{\mathsf{K}}}{3}][1-\alpha_{\mathsf{K}}]$	$[(\alpha_{\mathsf{H}}+\frac{\beta_{\mathsf{H}}}{3})(1-\alpha_{\mathsf{H}}-\frac{\beta_{\mathsf{H}}}{2})][(1-\alpha_{\mathsf{H}})\alpha_{\mathsf{H}}^{1-2\gamma_{\mathsf{H}}}]$
$(-1,1)$	$[1-\alpha_{\mathsf{K}}-\frac{\beta_{\mathsf{K}}}{3}][\alpha_{\mathsf{K}}]$	$[(1-\alpha_{\mathsf{H}}-\frac{\beta_{\mathsf{H}}}{3})(\alpha_{\mathsf{H}}+\frac{\beta_{\mathsf{H}}}{2})^{1-3\gamma_{\mathsf{H}}}][\alpha_{\mathsf{H}}(1-\alpha_{\mathsf{H}})]$
$(1,1)$	$[\alpha_{\mathsf{K}}+\frac{\beta_{\mathsf{K}}}{3}][\alpha_{\mathsf{K}}]$	$[(\alpha_{\mathsf{H}}+\frac{\beta_{\mathsf{H}}}{3})(\alpha_{\mathsf{H}}+\frac{\beta_{\mathsf{H}}}{2})^{1-3\gamma_{\mathsf{H}}}][\alpha_{\mathsf{H}}\alpha_{\mathsf{H}}^{1-2\gamma_{\mathsf{H}}}]$

Table 6.2: Probability of nodes 2 and 3's state configurations, specified by the cascade model K and the hGMM H.

The generative model's probability distribution function is described in Table 6.1

One can derive a set of equalities, such as $\Pr(a_2^2 = 1, a_3^2 = -1) = \Pr(a_2^2 = -1, a_3^2 = 1)$, from the generative model's probability distribution function described in Table 6.1 by solving for the parameter β. As a result, any model seeking to produce a probability distribution that matches the underlying distribution need satisfy these equality conditions. For simplicity, I drop the a notation and abbreviate $\Pr(x, y) = \Pr(a_2^2 = x, a_3^2 = y)$ henceforth.

I next focus on the aforementioned equality $\Pr(1,-1) = \Pr(-1,1)$. Applying this equality to the corresponding probability distribution of model K specified in Table 6.2 leads to $\alpha_{\mathsf{K}} = 0$. This consequently forces $\Pr_{\mathsf{K}}(1,-1)$ and $\Pr_{\mathsf{K}}(1,1)$ to be zero, even though these two cases are assigned non-zero probability in the generative model.[5] Thus, even when given a sufficiently large learning data set, one will never be able to learn a cascade model K on a graph with missing edges that optimally matches the underlying cascade process.

Unlike the cascade model K, the hGMM H specifies a probability distribution function that can both satisfy the generative model's derived equalities and still allow its parameters to obtain values that may help H capture the underlying cascade. In other words, for this particular case, one can construct an hGMM H that produces the same probability

[5] One can arrive at the same outcome by using a different equality induced by the generative model: $\Pr(-1,-1) = 1 - \sqrt{\Pr(1,1)}$.

distribution as the generative cascade model when given sufficient training data.

Note that the hGMM H has a higher degree of freedom than the cascade model K. However, the presence of additional parameters in H does not single-handedly determine the model's flexibility, and its modeling advantages; the positions of its parameters matter as well. For instance, if one omitted the parameter γ_{H} in two probabilities $\Pr_{\mathsf{H}}(-1, 1)$ and $\Pr_{\mathsf{H}}(1, -1)$, the equality $\Pr_{\mathsf{H}}(1, -1) = \Pr_{\mathsf{H}}(-1, 1)$ would force α_{H} to be zero. In fact, the inclusion of an additional term that involves node 3 and is governed by the parameter γ_{H} allows quantitative expression of the correlation between nodes 1 and 3, both of which contribute to the potential of node 2. Furthermore, one can always assign the following parameters $(\alpha_1, \beta_1, \alpha_{-1}, \beta_{-1}, \gamma)$ of any hGMM to zero in order to produce the same probability distribution as a cascade model for the same input graph. Thus, the above claims about the two models' capabilities for this example scenario generally holds true in larger systems.

6.3.3 Learning Graphical Structures

The structure-learning approach focuses on discovering unobserved connections using diffusion observation data. I consider two algorithms: one adapted from prior work under a different model, and the second a straightforward greedy learner introduced here.

6.3.3.1 `NetInf` and `NetInf′` Algorithms

Gomez-Rodriguez et al. (2010) employed the independent cascade model to capture information diffusion, and further assumed that every node becomes infected from exactly one other infected neighbor node, which allows them to view one diffusion instance as a directed tree. Given a graph structure G, let $\mathcal{T}$ be the set of all possible directed trees τ whose edges E_τ are a subset of E_G. A model M over G specifies the likelihood of a diffusion instance s on each tree $\tau \in \mathcal{T}$ as follows. I first describe the computation of $\Pr(i; s, \tau)$: the probability of observing node i becoming infected in s on tree τ. For each

node i such that there (uniquely) exists an edge $(j, i) \in \tau$, $\Pr(i; s, \tau)$ is defined as the probability of transmission from node j, infected at time t_j^*, to node i at time t_i^*, which is a function of $t_i^* - t_j^*$. Gomez-Rodriguez et al. (2010) introduced the `NetInf` algorithm that employs the power-law and exponential waiting time models in specifying $\Pr(i; s, \tau)$, which would put it at a disadvantage when given data from this study's generative cascade model. Therefore, the modified version `NetInf′` replaces the original definition in `NetInf` with $\Pr(i; s, \tau) = \beta(1 - \beta)^{(t_i^* - t_j^* - 1)}$ if $t_j^* < t_i^*$, and $\Pr(i; s, \tau) = 0$ otherwise. For each i that does not have an incoming edge in τ, the probability of observing node i's spontaneous infection is $\Pr(i; s, \tau) = \alpha$. Model M then interprets the probability of observing s on tree τ: $\Pr(s; \tau) = \prod_i \Pr(i; s, \tau)$. From this the algorithm approximates the probability of observing s on graph G as that of observing s on the maximum-likelihood tree τ_s: $\Pr(s \mid G) \approx \max_{\tau \in \mathcal{T}} \Pr(s; \tau)$.

Starting with some initial graph G, at each step, the `NetInf′` algorithm adds an edge (i, j) that maximizes the log likelihood of data S: $L_{tree}(S \mid G \cup (i, j)) = \sum_{s \in S} \log(\Pr(s \mid G \cup (i, j))$. As L_{tree} is monotone in graph size, this method would produce the complete graph without a limit K on the number of edges that can be added. Note that α and β here have the same interpretation as in the cascade model (6.1). Since `NetInf′` takes parameters α and β as input and does not learn them from data, I provide it with the parameter values used to generate the input data. I refer to the model learned by `NetInf′` as $\mathbf{netC}$.

6.3.3.2 `MaxLInf` Algorithm

I introduce a similar greedy algorithm `MaxLInf`, described in pseudocode below. The result $\mathbf{maxC}$ is a cascade model (6.2). Unlike `NetInf′`, the `MaxLInf` algorithmlearns the cascade model parameters α and β as well as the graphical structure G'. The algorithm employs as its objective function the average log likelihood (6.4) of diffusion instances S.

Note that `MaxLInf` does not require any predetermined constraint on the model's complexity, since adding more edges does not necessarily increase the likelihood L of input

Algorithm 2 `MaxLInf`

Input (G, S)
$G' \leftarrow G$
Learn (α, β) that maximizes $L^{\mathbf{maxC}}(S \mid G')$
repeat
  Find $(i, j) \notin G'$ maximizing $L^{\mathbf{maxC}}(S \mid G' \cup (i, j))$
  if $\Delta(i, j) = L^{\mathbf{maxC}}(S \mid G' \cup (i, j)) - L^{\mathbf{maxC}}(S \mid G') > 0$ **then**
    $G' \leftarrow G' \cup (i, j)$
    Learn (α, β) that maximizes $L^{\mathbf{maxC}}(S \mid G')$
  end if
until $\Delta(i, j) > 0$ or $\neg\exists(i, j) \notin G'$

diffusion observations.

6.3.3.3 Discussion

In the example of Section 6.2.1, these two learning algorithms may be able to discover the hidden edge $(1, 3)$, given sufficient observations. Since they are greedy, however, they may also add spurious edges that distort their views of the true network. For example, given some limited data set in which the case where node 4 was infected before node 3 occurs disproportionally frequently, they may greedily add $(3, 4)$ first to best explain the data, and stop before adding $(1, 3)$.

6.4 Empirical Study

The empirical study employs graphs of modest size to focus on investigating the relative ability of structure learning and graphical models to deal with unobserved connections. I empirically evaluate models $\mathbf{paG}$, $\mathbf{tabG}$, and $\mathbf{maxC}$ against baseline models $\mathbf{C}$ and $\mathbf{netC}$ on simulated data generated from the true cascade $\mathbf{C}^*$, with various graphical structures and experiment settings. I examine their detailed prediction performance based on the log likelihood L, aggregate prediction performance captured in the skew divergence D, and graph difference measures δ_- and δ_+.

6.4.1 Data Generation

The empirical study consists of two experiment groups: (A) stylized graphs of three to five nodes, and (B) larger graphs of size 30 and 100. In experiment group B, I employ the Erdos-Renyi model (ER) and the Barabasi-Albert preferential attachment model (PA) to generate random graphs G^* (Erdos and Renyi, 1959; Barabási and Albert, 1999). The ER model randomly creates an edge between each pair of nodes with equal probability, independently of the other edges. The PA model constructs graphs such that more connected nodes are more likely to receive new connections. I generate data from the $\mathbf{C}^*$ model on the graphs G^* with time horizon $T = 30$, using the α and β values that Stonedahl et al. (2010) identified from various real-world social networks at two different diffusion speeds (different values for α and β): normal and fast (viral).

Next, 50% of the edges from each G^* are uniformly randomly removed to generate the observable graph G, supplied as input to all the methods tested. Whereas Gomez-Rodriguez et al. (2010) developed and evaluated their original `NetInf` algorithm without graph input, the modified version `NetInf`$'$ is readily seeded with the partial structure G.

The log-likelihood function (6.4) is employed for training models in the methods studied here, except for `NetInf`$'$. Learning model parameters entails searching for settings that maximize the objective function using gradient descent with early stopping and random restarts.

6.4.2 Experiment Settings

Each result data point is averaged over 10 trials. The group A trials use 200 diffusion instances for training and 500 for testing. In experiment group B, I vary the number of diffusion traces for training from 25 to 100, and use 200 instances for testing in each trial. For clarity, the experiment result section presents the negated log likelihood measure $-L^M(S)$ for a model M; lower values indicate better performance. To assess aggregate infection statistics for model M, 200 diffusion traces of length $\hat{T} = 5$ are generated from the ini-

tial state of each given test instance s. From these I construct the conditional distribution $Q^M(c(a^{\hat{T}})/n \mid a^0)$. Given the distributions Q^M, one can calculate the aggregate prediction performance for model M, $D(Q^{\mathbf{C}^*}, Q^M \mid S)$. To establish statistical comparisons of two models M_1 and M_2, I further perform bootstrapped paired t-tests for both measures L and D. There are three different outcomes in comparison tables: M_1 outperforms M_2 with $p < 0.05$ (black), M_2 outperforms M_1 with $p < 0.05$ (white), and neither (gray). I investigate two variants of `NetInf'`, set to induce graphs with 50% (model $\mathbf{netC}$-small) and 100% (model $\mathbf{netC}$-large) more edges than the given graph G. In addition to trials with partially observed networks (partial), I include some fully observed cases (full) for calibration and sanity checking.

I label my experiments to indicate the following features: (i) coverage of the underlying graph by the given graph (partial or full), (ii) graph family (PA or ER), (iii) graph size, (iv) fast cascade speed in the generative model, and (v) number of input diffusion traces (in parenthesis).

6.4.3 Results

I present results on detailed prediction performance, measured by $-L$, for the 5-node case in Figure 6.3. I used the insights gained from analyzing the potential function parameters for $\mathbf{tabG}$ in the three-node and four-node cases to construct $\mathbf{paG}$, but do not report results from these cases here. I also omit comparisons of D, δ_-, and δ_+, which provide little differentiation among the models for such small networks.

Models $\mathbf{maxC}$, $\mathbf{tabG}$, and $\mathbf{paG}$ provide better detailed prediction performance than the baseline models $\mathbf{C}$ and $\mathbf{netC}$, when the input graph has missing edges. When the input graph is fully observed, $\mathbf{C}$ is the correct model, and as expected provides the best predictions. In addition, the hGMMs outperform $\mathbf{maxC}$ in the normal-spreading case but trail $\mathbf{maxC}$ in the fast-spreading scenario. In fact, I observe that $\mathbf{maxC}$ contains more correct edges (false negative $\delta_-(G^*, G') = 0.1$) in the fast spreading case than in the normal

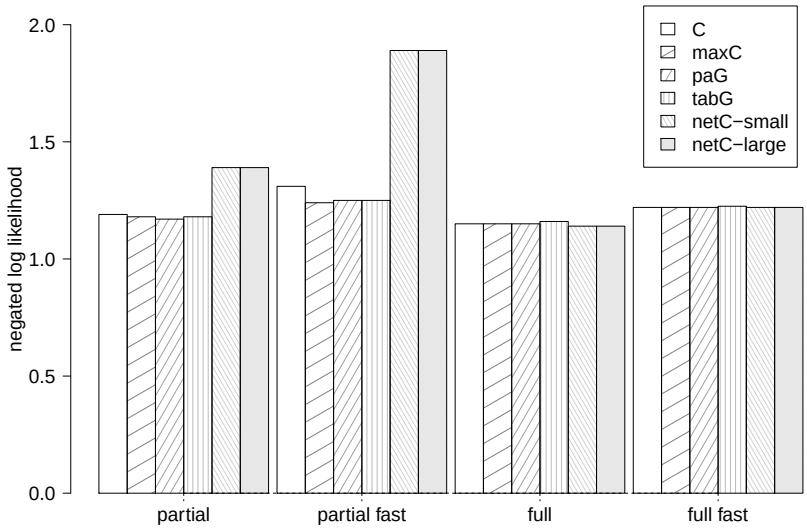

Figure 6.3: Detailed prediction performance for the 5-node graph (normal and fast spreading speeds).

	partial-5	partial-5 -fast	full-5	full-5-fast
maxC vs **C**				
tabG vs **C**				
paG vs **C**				
netC-small vs **C**				
netC-large vs **C**				
paG vs **maxC**				

Table 6.3: Pairwise comparisons of detailed predictions for the 5-node graph. For M_1 vs M_2 black indicates M_1 outperforms M_2 with $p < 0.05$, white the reverse, and gray no significant difference found.

case ($\delta_{-}(G^*, G') = 0.25$). As there are more spontaneous infections and fewer inter-node transmissions in the normal case, `MaxLInf` may too quickly add edges that help to explain spontaneous infections even though these edges are not present in the underlying graph. The statistical test results in Table 6.3 further cement confidence in the model comparisons above.

The results for larger graphs of 30 and 100 nodes in Figures 6.4 and 6.5 and Table 6.4 confirm that the proposed hGMM and `MaxLInf` approaches in general provide better predictions than $\mathbf{C}$ when there are missing edges. Since $\mathbf{paG}$ is more complex than the cas-

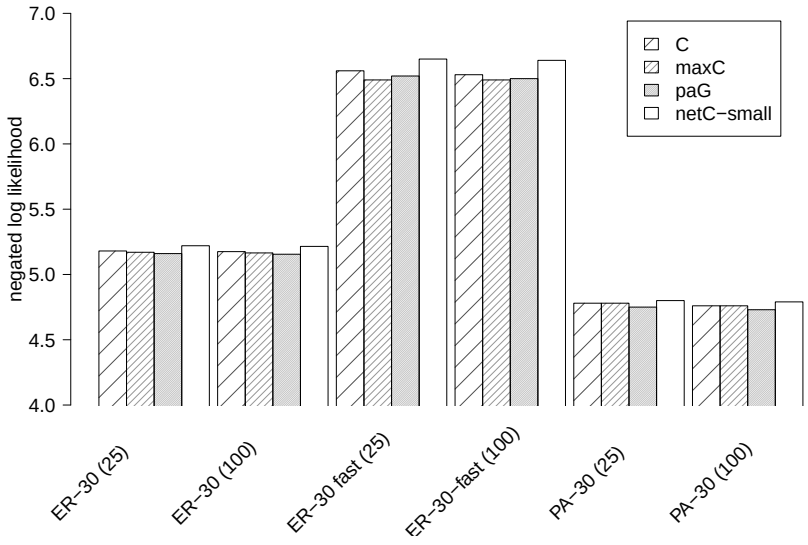

Figure 6.4: Detailed predictions in experiment group B for ER and PA 30-node graphs with different amounts of training data (numbers in parenthesis).

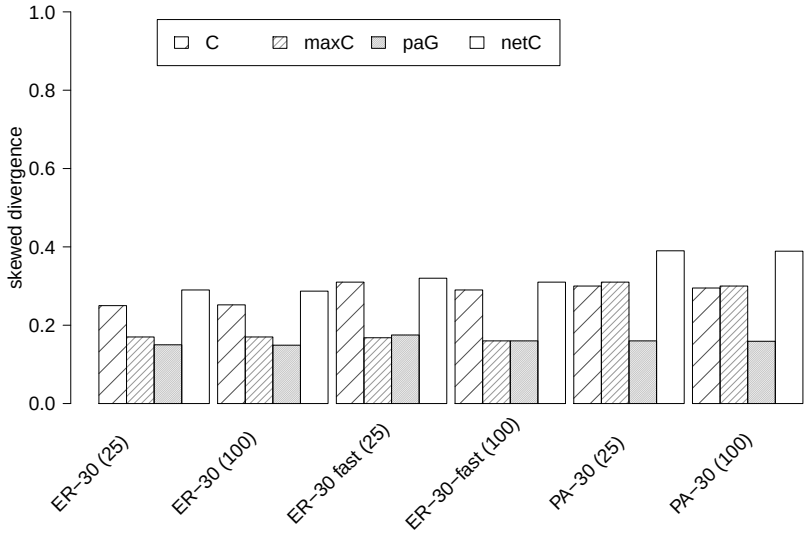

Figure 6.5: Aggregate predictions in experiment group B for ER and PA 30-node graphs with different amounts of training data (numbers in parenthesis).

cade models, **paG**'s performance improves as the amount of training data increases. With sufficient data, **paG** emerges as the best performer, exceeding **C** and notably **maxC** in both ER and PA graphs of 30 and 100 nodes. In fast diffusion cases, **paG**'s performance trails that of **maxC**. As for experiment group A, in group B, the cascade model **C** is the

Detailed Predictions	**maxC vs C**	**paG vs C**	**paG vs maxC**
partial-ER-30(25)			
partial-ER-30(100)			
partial-ER-30-fast(25)			
partial-ER-30-fast(100)			
partial-ER-100(25)			
partial-PA-30(25)			
partial-PA-30(100)			

Aggregate Predictions	**maxC vs C**	**paG vs C**	**paG vs maxC**
partial-ER-30(25)			
partial-ER-30(100)			
partial-ER-30-fast(25)			
partial-ER-30-fast(100)			
partial-ER-100(25)			
partial-PA-30(25)			
partial-PA-30(100)			

Table 6.4: Pairwise comparisons of detailed predictions (left) and aggregate predictions (right) for experiment group B for ER and PA graphs of 30 and 100 nodes with different amounts of training data (numbers in parenthesis). For M_1 vs M_2 black indicates M_1 outperforms M_2 with $p < 0.05$, white the reverse, and gray no significant difference found.

best model when the underlying graph is fully observed.

Model $\mathbf{maxC}$ outperforms the baseline $\mathbf{C}$ for ER graphs, but trails $\mathbf{C}$ for PA graphs. One may speculate that missing edges in PA graphs may be harder to learn, particularly for nodes with few connections. Figure 6.6 suggests that as the amount of training data increases, `MaxLInf` becomes more conservative in adding edges, reflected by its relatively constant false negative measure δ_- and decreasing false positive δ_+.

I do not show most of the results for `NetInf'`, as the models produced by this algorithm

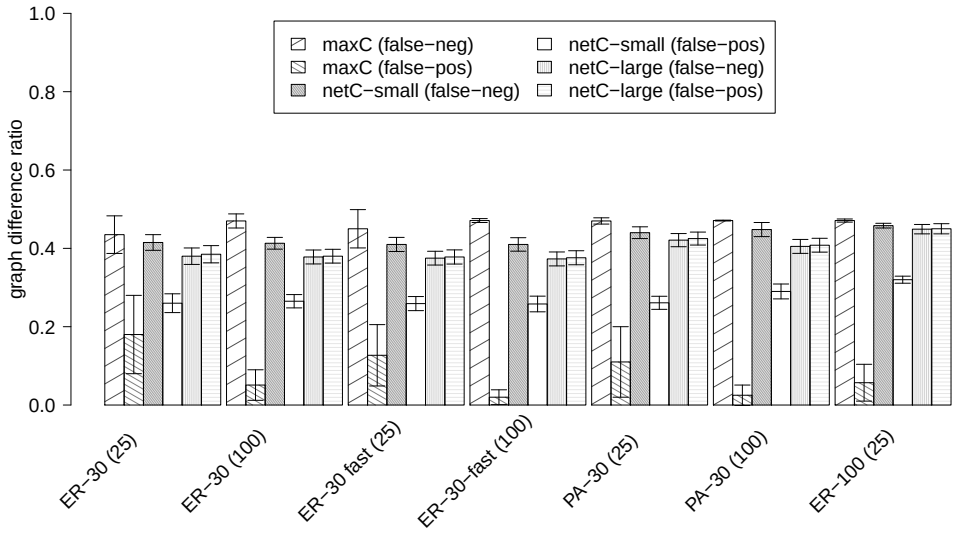

Figure 6.6: Graph difference measures in experiment group B for 30-node and 100-node graphs with different amounts of training data.

perform relatively poorly by the measures introduced in this study, despite the advantage of being given the true cascade model's parameter values. This discrepancy is likely a consequence of `NetInf′`'s use of an objective function L_{tree} at variance from this study's use of $L^M(S)$ for evaluation. The respective objective functions are applicable for two different types of network influence data: ordinal-time and snapshot (Cosley et al., 2010). The objective L_{tree} is suitable for ordinal-time data sets, which record the time of each individual node's change of state. The majority of data sets, however, store snapshots of the network state at various time periods, which align more directly with the objective $L^M(S)$. Future work could investigate the connection between these two objective functions, based on recent work exploring the correspondence between ordinal time and snapshot data (Cosley et al., 2010).

Even for ordinal data, L_{tree} is only an approximate measure of the likelihood of observed infection times, and thus does not exactly capture how information diffuses through connections. `NetInf′` also requires a predetermined constraint on the final graph's complexity. With respect to graph difference metrics, `NetInf′` is able to discover more missing edges than `MaxLInf`, resulting in a lower proportion of false negatives δ_- as depicted in

	5 (200)	5-fast (200)	ER (25)	ER-fast (25)
b_1 partial	2.37	1.65	4.68	1.44
b_1 full	0.051	0.038	0.24	0.01
b_{-1} partial	3.87	3.15	2.68	1.722
b_{-1} full	0.02	0.01	0.03	0.015

Table 6.5: Parameter comparisons in $\mathbf{paG}$ in experiment groups A and B (data amounts in parentheses).

Figure 6.6. However, `NetInf'` also adds significantly more spurious edges than `MaxLInf`, as reflected by δ_+. This problem worsens as `NetInf'` has more freedom to add new edges: $\mathbf{netC}$-large contains many more spurious edges than $\mathbf{netC}$-small, although $\mathbf{netC}$-large is basically given as input the exact number of unobserved connections. Moreover, whereas $\mathbf{maxC}$ can compensate for the learned graph's inaccuracy by fitting its parameters to data, $\mathbf{netC}$ has to use the fixed parameters from the generative cascade model, since the objective function L_{tree} is maximized when $\alpha = \infty$ and $\beta = 1$.

Learned parameters of $\mathbf{paG}$ can provide additional insights about the underlying network. In particular, analyzing $\mathbf{paG}$'s β_1 and β_{-1} may help to detect if the given network has unobserved edges. Intuitively, given a fully observed network, $\mathbf{paG}$ should use β solely in capturing information transmissions facilitated by the given network, and set β_1 and β_{-1} to near zero. When there exist missing edges, $\mathbf{paG}$ may assign higher values to β_1 and β_{-1} to capture how information diffuses among its seemingly unconnected neighbors. Let me define $b_1 = \beta_1/\beta$ and $b_{-1} = \beta_{-1}/\beta$. Table 6.5 shows that both b_1 and b_{-1} are notably higher in partially observed graphs than in fully observed structures, confirming this intuition. However, a more thorough examination using a wide range of measures is needed in order to draw a sufficiently confident connection between learned $\mathbf{paG}$ parameters and network structure.

`NetInf` was originally designed for a different evaluation measure, which accounts for

its relatively poor showing in this study. It remains possible that some ideas of that method could be incorporated in an algorithm that learns network structures with high predictive accuracy. Note that the inference complexity of $\mathbf{paG}$ grows exponentially with respect to the largest neighborhood's size. Although this result enables $\mathbf{paG}$ to model medium-size systems, such as academic co-authorship networks (Leskovec et al., 2009) and consensus dynamics experiments (Kearns et al., 2009), handling large social networks on the order of Facebook or Twitter remains a challenge. In one potential approach, one can exploit the potential function's generalized form, $\pi_i(a^t_{N_i} \mid a^t_{\Gamma_i})$, where the neighborhood size of $a^t_{N_i}$ dictates the inference complexity, while the role of $a^{t-1}_{\Gamma_i}$ in the model's complexity is negligible. In other words, one may choose to include only nodes whose states are most likely to be correlated in the within-time neighborhood N_i, while keeping the conditioning neighborhood Γ_i intact. Another approach is to apply the mean field method for approximate inference in graphical models, which in general generates less accurate approximations than the currently employed method, belief propagation (Weiss, 2001). However, the recent success of the mean field method in comparable settings with large social networks containing tens of thousands of users (Shi et al., 2009) suggests that its employment in GMMs can potentially provide reasonably accurate results and acceptable scalability in large domains.

In summary, if structure learning could recover the true network reliably, that would clearly be preferred under the assumption that the actual data is generated by the cascade model. The inevitable imperfection of structure learning, however, opens the door to alternative model forms, such as the history-dependent graphical multiagent models.

CHAPTER VII

Conclusion

I have introduced in this thesis a new probabilistic modeling framework for multiagent scenarios. Graphical multiagent models are essentially probabilistic graphical models in the context of multiagent systems: a graphical model expresses a joint probability distribution over agent behavior, exploiting sparseness in agent dependence to achieve tractability in representation and inference. The GMM formalism imposes no assumption about the underlying drivers of agent behavior and reasoning, and as such, can incorporate beliefs about agent reasoning derived from any agent theory or observational data. In time-variant settings, history-dependent GMMs extend the basic static form of GMMs to capture dynamic behavior through conditioning actions on some abstracted representation of history. Similar to static GMMs, history-dependent GMMs provide modelers with the flexibility to express directly agent dependencies resulting from various sources, such as history abstraction in the consensus dynamics scenario, and missing edges in the information diffusion scenario. As the advantages of a probabilistic graphical model mostly originate from its corresponding dependence graph structure, I have further examined the problem of learning dependence graph structures from data, and evaluated structure learning algorithms in specific multiagent scenarios. Empirical evidence suggests that differentiating probabilistic dependence structures from those that capture agent communication or interactions can potentially improve predictive power, as well as provide deeper insights on agent interactions

and reasoning.

7.1 Summary of Contributions

The flexibility and efficacy of both graphical multiagent models (static) and history-dependent graphical multiagent models (dynamic) are illustrated and supported by a series of computational studies, employing simulated and human-subject data in learning dependence graph structures, constructing graphical models, and making inferences and predictions about agent behavior.

7.1.1 Expressing Different Beliefs about Agent Reasoning

- In Section 3.1, I formally define and describe graphical multiagent models as probabilistic undirected graphical models in multiagent settings. By exploiting agent interdependence, GMMs achieve representational complexity that is exponential only with respect to the largest neighborhoods, but still has computational complexity exponential in the number of agents. That GMMs are basically probabilistic undirected graphical models allows one to take advantage of well established approximation methods in computing inferences by GMMs.

- The GMM potential functions can assume any functional forms, specified by any agent theory, and more importantly can be defined over not only one agent's actions but those of its dependence neighborhood. This capability thus provides the system modeler with great representational flexibility. The examples of potential functions outlined and discussed in Section 3.2 help to illustrate how different beliefs about agent reasoning, from game-theoretic solutions to information diffusion theory, can be expressed within the GMM framework.

- The ability to integrate multiple knowledge sources is demonstrated through the simulation study of technology upgrade in Sections 3.3 and 3.4: models based on game-

theoretic equilibrium and heuristic selection are employed to predict the outcome of a scenario where multiagent reinforcement learners make decisions on a network. The study finds that a model combining the two sources outperforms either source by itself, across a range of proposed combination methods: direct update, opinion pool, and mixing data. The knowledge integration capability of GMMs is a direct manifestation of their representational flexibility, thus bolstering the central argument for applying GMMs in multiagent system modeling.

7.1.2 Modeling Dynamic Behavior Dependence

- Chapter IV introduces history-dependent graphical multiagent models for dynamic scenarios. In Section 4.2 I outline the distinction between individual and joint behavior multiagent models. I further present arguments and illustrative examples that the incorporation of only some abstraction of history due to various observational and computational limitations induces action dependence beyond the capabilities of individual behavior models. The empirical study in Section 4.4 demonstrates the benefits of hGMMs in capturing multiagent behaviors in the example scenario of consensus dynamics. In particular, I compare a joint behavior hGMM JCM and an individual behavior hGMM ICM, which are designed to capture dynamic behavior sampled from a generative individual behavior model. Performance analysis of these two models demonstrates the benefit of hGMMs' joint-action modeling, even for predicting behavior that is known to be generated by independent choices of agents conditioning on common history. By controlling for the amount and granularity of action history available for these two models, I show that the greater the information loss is due to abstraction, the more JCM outperforms ICM in predicting system outcomes.

- Chapter VI investigates hGMMs' capabilities in modeling action dependence emerging from a different source: missing edges in the agent interaction graph. In particu-

lar, the problem is to model information diffusion on networks (interaction graphs). That some edges are missing or unobserved induces correlations among state changes, which can be effectively captured in the information diffusion hGMM $\mathbf{paG}$, as argued and illustrated in Section 6.3. This approach of directly modeling induced state correlations by employing $\mathbf{paG}$ is then evaluated against a different approach of learning missing connections for the standard independent cascade model, which is basically an individual behavior model. Experiments with different types and sizes of graph structures suggest that the hGMM approach generates predictions that are either more or comparably accurate than predictions generated by the greedy graph learning algorithm `MaxLInf` proposed in this study, and an adopted version of an existing algorithm `NetInf`. Moreover, the model $\mathbf{paG}$ itself provides an alternative to existing information diffusion models, as well as insights on which additional factors related to a neighborhood's joint states one may consider adding to a probabilistic model of information diffusion on networks.

7.1.3 Learning Both Dependence Graph Structure and Model Parameters from Observational Data

- The problem of learning graphical game models, which capture payoff dependencies among agents, is motivated and formulated in Section 5.1. I introduce formal definitions of the objective function and the three different evaluation metrics. I then present theoretical results establishing that the objective loss function increases monotonically with respect to the number of graph edges. This claim suggests the need for graph complexity constraints, effectively transforming the original problem into a constrained loss minimization problem. An empirical evaluation of four structure learning algorithms, branch-and-bound search with pruning, greedy loss minimization, and two versions of simulated annealing, on simulated graphical game data, reveals that a greedy approach often offers better time-performance tradeoff.

- An in-depth study of the consensus dynamics scenario, using human-subject experiment data described in Section 5.2, illustrates the facility of GMMs in capturing dynamic behavior, and the viability of learning GMM structure and parameters from data. In particular, I propose an algorithm that simultaneously greedily adds edges to the dependence graph and tunes model parameters, aiming at maximizing the likelihood of training data. This study proposes a joint behavior hGMM eJCM and a corresponding individual behavior hGMM eICM, and furthermore empirically compares these two models with two existing models PRM and sPRM. In particular, the focus is on learning within-time edges in eJCM, as other models assume no within-time edges by definition. eJCM outperforms the individual behavior models eICM, PRM, and sPRM, as well as a version of eJCM that does not differentiate within-time and across-time edges, in predicting human-subject behaviors. This outcome demonstrates the feasibility of learning both dependence graph structure and model parameters, using the greedy algorithm.

- Modeling multiagent behavioral patterns of human-subject experiments further stresses the need to induce dependence structure from data, and yields insights about the relationships between the structure of network interactions and probabilistic dependency. Analysis shows that relationships between agents favoring the same consensus color outnumber other relationships in the learned graphs. That certain relationships or connections between agents are more important to probabilistic modeling than others further strengthens the argument for differentiating dependence graph structures and observation or interaction graph structures in order to gain insights about various relationships between heterogeneous agents.

7.2 Model Construction Guidelines

When tasked to model a multiagent system, the modeler can generate substantial gains in performance and efficiency by incorporating the aforementioned findings in each step of the model construction process. As dynamic scenarios are more frequently studied, in this section I particularly focus on history-dependent GMMs.

First of all, the system modeler needs to pick the appropriate balance between prediction performance and model complexity, within given computational limitations and data availability constraints. In particular, the complexity of an hGMM depends on two factors: (i) the size of the largest dependence neighborhood N, and (ii) the history abstraction function's complexity, governed by its functional form and history length h. As demonstrated in Chapters IV and V, a complexity increase of dependence neighborhoods N can compensate, in terms of prediction performance, for a decrease in the abstraction function's complexity, which is often restricted by limited or partial observation data given as input. Note that individual behavior hGMMs have the least complex dependence neighborhoods, each of which contains only one agent. At the same time, the learned model's prediction performance improves as its complexity rises, up to a point where overfitting starts to erode performance. As a result, when constructing a history-dependent GMM, the system modeler needs to address the following questions:

1. What is the maximum performance loss that one is willing to accept in order to achieve reasonable efficiency in inference?

2. Given the amount, scope, and completeness of input data employed in the history abstraction function, how complex should the dependence graph be in order to generate acceptably accurate predictions?

Figure 7.1 illustrates the aforementioned relationships between complexity, performance, and data availability within the framework of history-dependent GMMs.

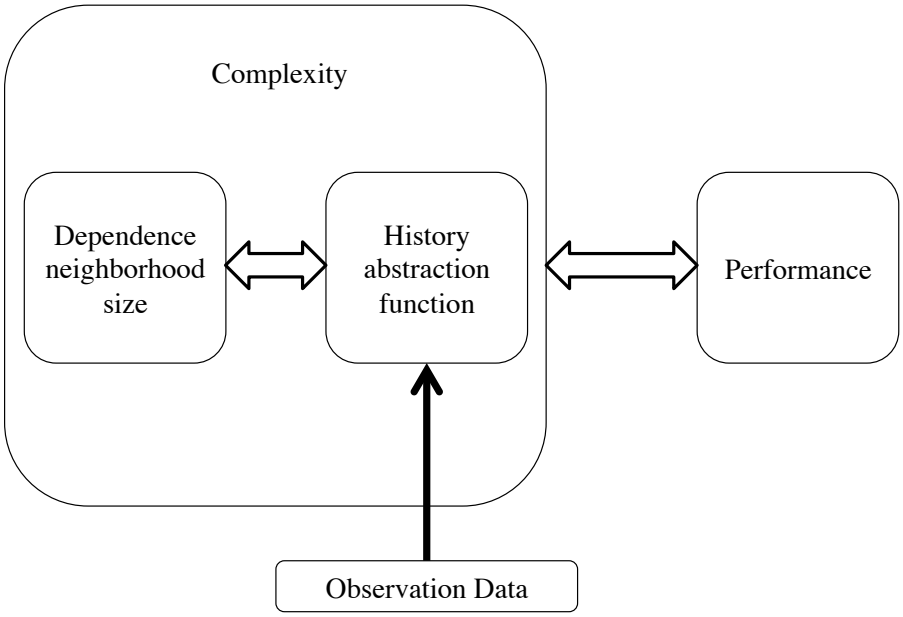

Figure 7.1: Relationships between complexity, performance, and data availability in the construction of history-dependent GMMs. Dependence neighborhood size and history abstraction function determine the model's complexity. Two-headed fat arrows indicate the reverse relationships among the two connected factors: complexity versus performance, and dependence neighborhood size versus history abstraction function (given a predetermined performance goal). The scope and completeness of observation data often directly dictate the form and complexity of the history abstraction function.

After answering these high-level design questions, the system modeler needs to make several decisions about the model's details, as summarized in Figure 7.2. In particular, the next step involves specifying two main components of an hGMM: neighborhood potential function and probabilistic dependence graph structure. There are no systematic guidelines for defining the functional form of neighborhood potentials, which for the most part requires domain-specific tuition and background knowledge. However, one can learn the function's parameters from input action data by using any appropriate machine learning technique. Chapter V suggests that even if given some graphical structure representing agent interactions, the system modeler may want to learn a different probabilistic de-

pendence graph from action data. Moreover, the size of most multiagent modeling problems not only raises the need for a compact dependence graphical representation, but also prompts one to employ approximation techniques for inference in probabilistic graphical models.

Along with action data, the modeler often has access to a graph structure representing some observed interactions and communications between agents, such as face-to-face meetings, emails, and online chats. One may simply employ this graph structure as the modeler's probabilistic dependence graph, instead of learning the dependence graph from data, even when the given graph is only partially observed, as demonstrated in Chapter VI. However, its more effective usage usually directly involves the potential function: the observation graph structure provides not only a reasonable approximation of the conditioning sets, but also valuable intuition about different terms of the potential function and their interactions, as illustrated in Chapters IV and V.

7.3 Future Directions

That GMMs and hGMMs are probabilistic graphical models allows one to take advantage of many established graphical model inference and learning algorithms. Extending hGMMs to a more comprehensive dynamic presentation that supports not only forward but also backward inference to reason about unobserved past states can potentially open up the entire repertoire of graphical model methods for hGMM related tasks. Such a development will allow the inclusion of hidden variables (unobserved agent actions) in hGMMs, and thus broaden the appeals of the GMM framework in many real-world scenarios where incomplete observation data are the norm.

Due to its generality and relatively low complexity, belief propagation is employed throughout this thesis for approximating inferences in GMMs and hGMMs. However, the need for modeling very large multiagent systems, such as online social networks, prompts experimentation with less accurate but more efficient approximation methods. For instance,

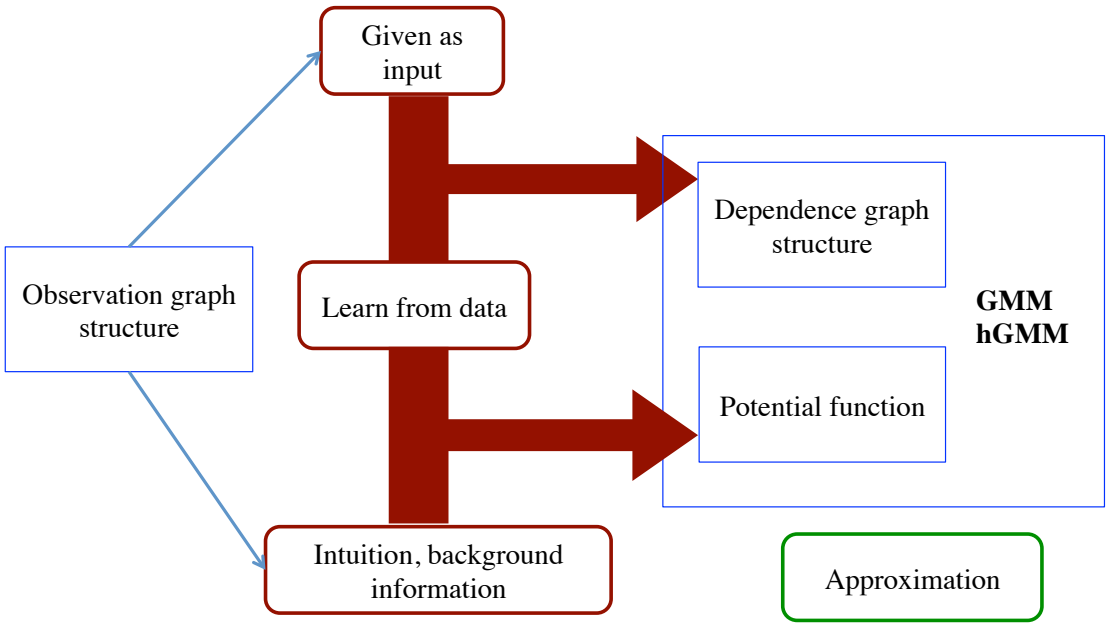

Figure 7.2: Highlights of the main decisions that require the system modeler's attention during the model construction process.

one may consider employing mean field methods, which in general generate less accurate approximations and are highly dependent on problem specifics, but are simple to implement and fast in producing approximate inference.

The importance of differentiating dependence graph structures from input structures of agent interaction or observation, as emphasized in this work, can potentially motivate more studies on the roles of interaction and observation edges in probabilistic inference. The study presented in this thesis on learning dependence structures from data can serve as a starting point for examining different types of graph edges in real-world multiagent systems. Moreover, there is room for greater improvement on the graph analysis of learned dependence structures presented in Section 5.2.3. A more systematic and comprehensive

toolbox for dissecting graphical structures of dynamic behavior and assessing model learning algorithmsÕ contributions will lend needed support to the investigations of many different dependence edge types. Analysis of these different edge types in turn can further provide insights on the nature of reasoning and interactions among humans and organizations.

Another research direction is to improve the learning algorithm for individual behavior models, by replacing the maximum-degree constraint with a cross-validation condition that can better help avoid over-fitting. Given the formalism's generality, it is promising to apply the graphical multiagent model technique to similar problem domains, such as graph coloring, where agents must coordinate their actions or make collective decisions while only communicating with their neighbors, as well as large network scenarios, such as social networks and Internet protocols.

One interesting application for the GMM framework is to study the problem of modeling network formation: identifying and analyzing the factors that drive the appearance and disappearance of nodes and edges in a network that captures relationships between people or organizations. In particular, a network formation model often seeks to achieve either one or ideally both of the following goals: (i) predict future network configurations given past observations, and (ii) complies with statistics observed in many real-world networks, such as node degree distribution and clustering coefficient. Although many have addressed this problem from different angles (Leskovec et al., 2007b; Backstrom and Leskovec, 2011; Christakis et al., 2010; Huang et al., 2008; Pemantle and Skyrms, 2004; Skyrms and Pemantle, 2000; Jackson and Watts, 2002), none has been able to achieve both goals satisfactorily. Statistical modeling approaches are mostly unconcerned with the underlying drives of edge formation decisions, and usually assume individual behavior. Game-theoretic models often impose unrealistic assumptions about the utility agents receive from establishing new connections and about their reasoning. In addition, the size of these problem domains also mounts a challenge to modeling efforts.

By employing the graphical multiagent model framework, one may be able to incorporate strategic reasoning elements from the game-theoretic models, while leveraging the flexibility of the potential functions in graphical multiagent models to avoid the game-theoretic approach's limitations, such as perfect rationality. Moreover, GMMs' ability to express joint action dependence can potentially help to model edge formation decisions, which are often correlated and coordinated in various real-world network scenarios. For example, consider the problem of modeling co-authorship networks, in which an edge between two nodes (researchers) signals that they have written and published a research paper together in the past (Leskovec et al., 2007b; Huang et al., 2008). Researchers are often connected with people who work on similar research topics or are in the same research lab. Thus, researchers who are in the same neighborhood or close together on the co-authorship network tend to make somewhat coordinated decisions about who they would like to collaborate with and what projects they are interested in. As a result, the system modeler may desire to employ a joint behavior model, instead of an individual behavior model, to capture action correlations. Note that unlike the multiagent scenarios studied in this thesis, action correlations here do not come from history abstraction or missing edges, but result directly from agent reasoning. In addition, more advanced and efficient statistical model construction and inference techniques will be needed to learn models of very large networks from observational data.

7.4 Concluding Remarks

Multiagent research has made great progress in modeling and analyzing increasingly large and complex systems of heterogeneous and autonomous agents. Real-world systems, such as online social networks, corporate alliances, and routing networks, often exhibit locality structures that allow efficient modeling representations and present insights about the foundation of their interactions. The tasks of analyzing and inferring about network interactions require both efficient computer algorithms on large data sets, and theories and

models from various fields, ranging from economics and sociology to industrial engineering and statistics. The need to study large complex networks present in every aspect of our life will accelerate the integration of artificial intelligence with these fields in creating more efficient and effective tools for modeling and analyzing network activities. The research in this thesis contributes to this goal of understanding and predicting social, technological, and economic network behavior by addressing the problem of compactly and accurately modeling multiagent behavior with a focus on understanding the cost and benefit reasoning that drives agent behavior. With the proposed graphical multiagent model framework, I strive to contribute to the integration of strategic behavior modeling techniques from the fields of multiagent systems and economics with compact probabilistic models from statistics that can efficiently extract behavior patterns from massive amounts of network data, for the goal of understanding the world's many fast-changing and complex multiagent systems.

BIBLIOGRAPHY

BIBLIOGRAPHY

Adar, E. and Adamic, L. A. (2005). Tracking information epidemics in blogspace. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 207–214, Compiegne, France.

Backstrom, L. and Leskovec, J. (2011). Supervised random walks: Predicting and recommending links in social networks. In *Fourth ACM International Conference on Web Search and Data Mining*, pages 635–644, Hong Kong.

Bakshy, E., Karrer, B., and Adamic, L. (2009). Social influence and the diffusion of user-created content. In *Tenth ACM Conference on Electronic Commerce*, pages 325–334, Stanford, California.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Broadway, J. Y., Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2000). Generalized belief propagation. In *Thirteenth Annual Conference on Advances in Neural Information Processing Systems*, pages 689–695, Denver.

Camerer, C. and Ho, T. (1999). Experienced-weighted attraction learning in normal form games. *Econometrica*, 67(4):827–874.

Chakrabarti, D., Leskovec, J., Faloutsos, C., Madden, S., Guestrin, C., and Faloutsos, M. (2007). Information survival threshold in sensor and P2P networks. In *Twenty-Sixth IEEE International Conference on Computer Communications*, pages 1316–1324, Anchorage.

Chickering, D. M. (2002). Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498.

Choudhury, M. D., Sundaram, H., John, A., and Seligmann, D. (2008). Dynamic prediction of communication flow using social context. In *Nineteenth ACM Conference on Hypertext and Hypermedia*, pages 49–54, Pittsburgh.

Christakis, N. A., Fowler, J. H., Imbens, G. W., and Kalyanaraman, K. (2010). An empirical model for strategic network formation. Technical report, National Bureau of Economic Research.

Cosley, D., Huttenlocher, D., Kleinberg, J., Lan, X., and Suri, S. (2010). Sequential influence models in social networks. In *Fourth International AAAI Conference on Weblogs and Social Media*, pages 26–33, Washington, DC.

Dagum, P. and Luby, M. (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153.

Daskalakis, C. and Papadimitriou, C. H. (2006). Computing pure Nash equilibria in graphical games via Markov random fields. In *Seventh ACM Conference on Electronic Commerce*, pages 91–99, Ann Arbor, Michigan.

De Choudhury, M., Mason, W. A., Hofman, J. M., and Watts, D. J. (2010). Inferring relevant social networks from interpersonal communication. In *Nineteenth International Conference on World Wide Web*, pages 301–310, Raleigh, NC.

Duong, Q., Vorobeychik, Y., Singh, S., and Wellman, M. P. (2009). Learning graphical game models. In *Twenty-First International Joint Conference on Artificial Intelligence*, pages 116–121, Pasadena, California.

Duong, Q., Wellman, M., Singh, S., and Vorobeychik, Y. (2010). History-dependent graphical multiagent models. In *Ninth International Conference on Autonomous Agents and Multiagent Systems*, pages 1215–1222, Toronto, Canada.

Duong, Q., Wellman, M. P., and Singh, S. (2008). Knowledge combination in graphical multiagent models. In *Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 153–160, Helsinki, Finland.

Duong, Q., Wellman, M. P., and Singh, S. (2011). Modeling information diffusion in networks with unobserved links. In *Third IEEE International Conference on Social Computing*, pages 362–369, Boston.

Duong, Q., Wellman, M. P., Singh, S., and Kearns, M. (2012). Learning and predicting dynamic networked behavior with graphical multiagent models. In *Eleventh International Conference on Autonomous Agents and Multiagent Systems*, pages 441–448, Valencia, Spain.

Erdos, P. and Renyi, A. (1959). On random graphs. i. *Publicationes Mathematicae*, 6(290-297):156.

Ficici, S. G., Parkes, D. C., and Pfeffer, A. (2008). Learning and solving many-player games through a cluster-based representation. In *Twenth-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 187–195.

Friedman, N. (1998). The Bayesian structural EM algorithm. In *Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 129–138, Madison, Wisconsin.

Fudenberg, D. and Levine, D. K. (1998). *The Theory of Learning in Games*. MIT Press.

Gal, Y. and Pfeffer, A. (2008). Networks of influence diagrams: A formalism for reasoning about agents' decision-making processes. *Journal of Artificial Intelligence Research*, 33(1):109–147.

Gao, X. and Pfeffer, A. (2010). Learning game representations from data using rationality constraints. In *Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence*, pages 185–192, Corvallis, Oregon.

Ghahramani, Z. (1998). Learning dynamic Bayesian networks. *Adaptive Processing of Sequences and Data Structures*, 1387:168–197.

Goldenberg, J., Libai, B., and Muller, E. (2001). Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3):211–223.

Gomez-Rodriguez, M., Balduzz, D., and Scholkopf, B. (2011). Uncovering the temporal dynamics of diffusion networks. In *Twenty-Eighth International Conference on Machine Learning*, Bellevue, Washington.

Gomez-Rodriguez, M., Leskovec, J., and Krause, A. (2010). Inferring networks of diffusion and influence. In *Sixteenth ACM International Conference on Knowledge Discovery and Data Mining*, pages 1019–1028, Washington, DC.

Granovetter, M. (1978). Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–43.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243.

Hennes, D., Tuyls, K., and Rauterberg, M. (2009). State-coupled replicator dynamics. In *Eighth International Conference on Autonomous Agents and Multiagent Systems*, pages 789–796, Budapest.

Huang, J., Zhuang, Z., Li, J., and Giles, C. L. (2008). Collaboration over time: characterizing and modeling network evolution. In *First International Conference on Web Search and Data Mining*, pages 107–116, Stanford, CA.

Jaakkola, T., Singh, S., and Jordan, M. I. (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In *Advances in Neural Information Processing Systems 7*, pages 345–352.

Jackson, M. O. and Watts, A. (2002). The evolution of social and economic networks. *Journal of Economic Theory*, 106(2):265–295.

Jensen, C., Kjaerulff, U., and Kong, A. (1995). Blocking-Gibbs sampling in very large probabilistic expert systems. *International Journal of Human Computer Studies*, 42(6):647–666.

Jensen, F. and Nielsen, T. (2007). *Bayesian Networks and Decision Graphs*. Springer Verlag.

Jiang, A., Leyton-Brown, K., and Bhat, N. (2011). Action-graph games. *Games and Economic Behavior*, 71(1):141–173.

Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Kakade, S., Kearns, M., Langford, J., and Ortiz., L. (2003). Correlated equilibria in graphical games. In *Fourth ACM Conference on Electronic Commerce*, pages 42–47, San Jose, CA.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

Kanazawa, K. and Dean, T. (1989). A model for projection and action. In *Eleventh International Joint Conference on Artificial Intelligence*, pages 985–990, Detroit.

Kappen, H. J. and Rodríguez, F. B. (1997). Mean field approach to learning in Boltzmann machines. *Pattern Recognition Letters*, 18:1317–1322.

Kearns, M. (2002). Computational game theory: A tutorial. `http://www.cis.upenn.edu/~mkearns/nips02tutorial/nips.pdf`. Presented at the *Neural Information Processing Systems* Conference.

Kearns, M. (2007). Graphical games. In Nisan et al. (2007), pages 159–180.

Kearns, M., Judd, S., Tan, J., and Wortman, J. (2009). Behavioral experiments on biased voting in networks. *Proceedings of the National Academy of Sciences*, 106(5):1347–1352.

Kearns, M., Littman, M. L., and Singh, S. (2001). Graphical models for game theory. In *Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 253–260, Seattle.

Kearns, M. and Wortman, J. (2008). Learning from collective behavior. In *Twenty-First Annual Conference on Learning Theory*, pages 99–110, Helsinki.

Kempe, D., Kleinberg, J., and Éva Tardos (2003). Maximizing the spread of influence through a social network. In *Ninth ACM International Conference on Knowledge Discovery and Data Mining*, pages 137–146, Washington.

Kindermann, R. and Shell, J. L. (1980). *Markov random fields and their applications*. American Mathematical Society.

Kleinberg, J. (2007). Cascading behavior in networks: Algorithmic and economic issues. In Nisan et al. (2007), pages 613–632.

Koivisto, M. and Sood, K. (2004). Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Koller, D. and Milch, B. (2003). Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45:181–221.

Krebs, V. (2002). Internet industry partnerships. `http://www.orgnet.com/netindustry.html`.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computation with probabilities on graphical structures and their applications to expert systems. *Journal of the Royal Statistics Society*, 50(2):157–224.

Lee, L. (1999). Measures of distributional similarity. In *Thirty-Seventh Meeting of the Association for Computational Linguistics*, pages 25–32, College Park, MD.

Leskovec, J., Adamic, L., and Huberman, B. (2007a). The dynamics of viral marketing. *ACM Transactions on the Web*, 1(1):5–43.

Leskovec, J., Kleinberg, J., and Faloutsos, C. (2007b). Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):2.

Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. (2009). Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123.

Mooij, J. M. (2008). libDAI 0.2.2: A free/open source C++ library for discrete approximate inference methods.

Mookherjee, D. and Sopher, B. (1994). Learning behavior in an experimental matching pennies game. *Games and Economic Behavior*, 7(1):62 – 91.

Murphy, K. (2001). An introduction to graphical models. Technical report, University of Britisih Columbia.

Neal, R. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical report, University of Toronto, Department of Computer Science.

Newman, M. E. J. (2003). Mixing patterns in networks. *Physical Review E*, 67(2).

Nielsen, J. D., Kočka, T., and Pena, J. M. (2002). On local optima in learning Bayesian networks. In *Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 435–442, Alberta, Canada.

Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V., editors (2007). *Algorithmic Game Theory*. Cambridge University Press.

Nudelman, E., Wortman, J., Shoham, Y., and Leyton-Brown, K. (2004). Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 880–887.

Pearl, J. (2000). *Causality Models, Reasoning, and Inference*. Cambridge University Press.

Pemantle, R. and Skyrms, B. (2004). Network formation by reinforcement learning: the long and medium run. *Mathematical Social Sciences*, 48(3):315–327.

Pennock, D. M. and Wellman, M. P. (2005). Graphical models for groups: Belief aggregation and risk sharing. *Decision Analysis*, 2:148–164.

Rosenthal, R. (1973). A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67.

Schmidt, M., Niculescu-Mizil, A., and Murphy, K. (2007). Learning graphical model structure using L1-regularization paths. In *Twenty-Second National Conference on Artificial Intelligence*, pages 1278–1283.

Shi, X., Zhu, J., Cai, R., and Zhang, L. (2009). User grouping behavior in online forums. In *Fifteenth ACM International Conference on Knowledge Discovery and Data Mining*, pages 777–786, Paris.

Shoham, Y. and Tennenholtz, M. (1997). On the emergence of social conventions: Modeling, analysis, and simulations. *Artificial Intelligence*, 94:139–166.

Silander, T. and Myllymaki, P. (2006). A simple approach for finding the globally optimal Bayesian network structure. Cambridge, MA.

Skyrms, B. and Pemantle, R. (2000). A dynamic model of social network formation. *Proceedings of the National Academy of Sciences*, 97(16):9340.

Smyth, P., Heckerman, D., and Jordan, M. I. (1997). Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, 9(2):227–269.

Song, X., Chi, Y., Hino, K., and Tseng, B. L. (2007). Information flow modeling based on diffusion rate for prediction and ranking. In *Sixteenth International World Wide Web Conference*, pages 191–200, Banff.

Stonedahl, F., Rand, W., and Wilensky, U. (2010). Evolving viral marketing strategies. In *Twelfth Conference on Genetic and Evolutionary Computation*, pages 1195–1202, Portland, OR.

Tuyls, K. and Parsons, S. (2007). What evolutionary game theory tells us about multiagent learning. *Artificial Intelligence*, 171:406–416.

Vorobeychik, Y. and Wellman, M. P. (2006). Mechanism design based on beliefs about responsive play (position paper). In *ACM EC-06 Workshop on Alternative Solution Concepts for Mechanism Design*, Ann Arbor, MI.

Weiss, Y. (2001). Comparing the mean field method and belief propagation for approximate inference in MRFs. In Opper, M. and Saad, D., editors, *Advanced Mean Field Methods: Theory and Practice*. MIT Press.

Wellman, M. P. (2006). Methods for empirical game-theoretic analysis. In *Twenty-First National Conference on Artificial Intelligence*, pages 1552–1555, Boston, MA.

Wolpert, D. H. (2006). Information theory: The bridge connecting bounded rational game theory and statistical physics. In *Complex Engineered Systems*. Springer.

Xing, E. P., Jordan, M. I., and Russell, S. (2003). A generalized mean field algorithm for variational inference in exponential families. In *Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 583–591, Acapulco.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Generalized belief propagation. *Advances in Neural Information Processing Systems*, 13:689–695.

Zeng, Y., Doshi, P., and Chen, Q. (2007). Approximate solutions to dynamic interactive influence diagrams using model clustering. In *Twenty-Second Conference on Artificial Intelligence*, pages 782–787, Vancouver, Canada.