NOVEL TASK DECOMPOSITION AND AGGREGATION
METHODS FOR KNOWLEDGE DISCOVERY IN MULTI-AGENT SYSTEMS

A Dissertation in
Electrical Engineering
by
Aditya Kurve

© 2013 Aditya Kurve

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2013
The dissertation of Aditya Kurve was approved* by the following:

George Kesidis  
Professor of Electrical Engineering and Computer Science and Engineering  
Dissertation Advisor, Chair of Committee

David J. Miller  
Professor of Electrical Engineering

John Doherty  
Professor of Electrical Engineering

Christopher Griffin  
Head of Cyber-Analytics and Operations Research at ARL

Jason Morton  
Assistant Professor of Statistics

Aydin Kultegin  
Professor of Electrical Engineering  
Head of the Department of Electrical Engineering

*Signatures on file in the Graduate School.
Abstract

Multi-agent systems allow us to study and co-ordinate autonomous processes or agents for achieving macroscopic system-level goals. Agents are typically characterized by self-organizing behavior with heterogeneity in their individual goals, local information states or hypotheses. This thesis studies two domain specific problems in multi-agent systems: task decomposition and task aggregation. The former deals with suitably tasking the agents to achieve system level goals under the constraint that each agent has limited knowledge of the current state of the system. The latter studies the problem of combining the results of the agents’ tasks by taking into account the heterogeneity in agent-level parameters and their hypotheses or perceptions. The first part of this thesis deals with an agent-level decomposition of a task for optimizing clusters in super-peer based peer-to-peer (P2P) systems for efficient knowledge discovery via reduction in the average query resolution time. In this scenario, peers acting as independent agents are suitably incentivized in order to achieve this system-level objective. In particular, each peer chooses a super-peer using a game-theoretic cost function based on its limited “scope of view” and the communal goal is the minimization of average query resolution time. The resulting self-organizing system of peers attempts to “locally” solve a graph partitioning problem, thereby, leading to a guaranteed equilibrium state under the assumption of static network parameters, or at least, guaranteeing reduction in transitions without such an assumption. In the second part of this thesis, we study novel methods of aggregating agent-level decisions for a communal learning task in a crowdsourcing domain. Herein, agents are human annotators who expedite the annotation of large databases and make individual inferences on a subset of tasks assigned to them. In this case, the goal is to fuse or aggregate their responses in order to learn the ground truth values. We present two approaches to crowd aggregation in the case of multicategorical annotation tasks. Firstly, we present a stochastic generative model of a worker’s response to the an-
notation tasks that uses a novel comparative paradigm of the worker’s skill and the difficulty level of the task. Secondly, we propose a model-independent weighted plurality-based aggregation rule that uses novel weight energy constraints. Both these methods defeat the “tyranny of the masses”, i.e., they exploit the expertise of a minority subset of highly skilled workers in a crowd of mostly low-skilled workers. We also model both “naively” malicious as well as “strategic” adversarial agent behavior. The third and the final part of the thesis studies the detection of sybils in a peer-to-peer (P2P) system, which typically uses distributed reputation and referral mechanisms to enforce fair content distribution across peers. Each peer is limited in terms of the knowledge of the reputation scores of other peers and hence centralized graph-based search is not suitable. Under such constraints, we propose a distributed sparse-cut detection algorithm based on a novel hierarchical implementation of Karger’s min-cut algorithm on a trust (or reputation) graph along with a local “sparsity” metric that measures the confidence of presence of a sybil cluster.
Table of Contents

List of Figures viii
List of Tables x
Acknowledgments xi

Chapter 1
Introduction to Multi-agent Systems 1
1.1 Intelligent Agents . 2
1.2 Challenges in Multi-agent Systems 2
1.3 Agent Architectures 3
1.4 Agent Utility 4
1.5 Multi-agent Interactions 4
   1.5.1 Nash Equilibrium 4
   1.5.2 Convergence to Nash Equilibria and Potential Games 5
1.6 Agent-based Modeling 6
1.7 Multi-agent Learning 6
1.8 Summary of Research Contributions 8

Chapter 2
Background on Machine Learning 10
2.1 Supervised, Unsupervised and Semi-supervised Learning 10
2.2 Generative and Discriminative Classification 11
2.3 Generative Probabilistic Modeling 13
2.4 Incomplete Data and EM algorithm 15
Chapter 3
Efficient Query Resolution by Optimization of Cluster Formation in Super-Peer Networks

3.1 Introduction ................................................. 17
3.2 Background .................................................. 20
3.3 System Overview ............................................. 23
3.4 A Local Cost Function ...................................... 25
  3.4.1 Nash Equilibrium ...................................... 26
3.5 An Alternative Cost Framework ............................ 27
  3.5.1 Comparison of the Two Cost Frameworks .......... 29
3.6 Local Incentive-based Algorithm .......................... 33
  3.6.1 Convergence ........................................... 35
3.7 P2P File Sharing Experiment .............................. 36
3.8 Numerical Results .......................................... 39

Chapter 4
Multicategory Crowdsourcing Accounting for Plurality in Worker Skill and Intention and Task Difficulty

4.1 Crowdsourcing: An Introduction ........................... 45
4.2 Generative Semisupervised Modeling
  4.2.1 Notation ................................................. 50
  4.2.2 Stochastic Generation Model .......................... 50
  4.2.3 Worker Types ........................................... 51
    4.2.3.1 Honest Workers .................................. 51
    4.2.3.2 Simple Adversarial Workers ...................... 51
  4.2.4 Incomplete, Complete and Expected Complete Data Log Likelihood ........................................... 52
  4.2.5 The Generalized EM (GEM) Algorithm ............... 53
  4.2.6 Unsupervised GEM ..................................... 56
4.3 Energy-Constrained Weighted Plurality Aggregation ........ 56
  4.3.1 From simple plurality to weighted plurality voting .. 57
  4.3.2 Accounting for Adversaries ........................... 59
    4.3.2.1 Introducing Binary Parameters to Represent Intention ........................................ 60
    4.3.2.2 Negative Weights Signify Adversaries .......... 61
  4.3.3 Semisupervised Case ................................... 63
4.4 Experiments ................................................ 64
  4.4.1 Experiments with Synthetic Data ...................... 64
  4.4.2 Simulating a Crowd using an Ensemble of Classifiers . 68
## List of Figures

2.1 Decision boundary directly inferred from the training data using logistic regression. ........................................... 12
2.2 Data labeled (using different colors) by a Gaussian mixture model. ......................................................... 13
2.3 A graphical model of health diagnosis for predicting diseases ................................................................. 14
2.4 A graphical model of a Naive Bayes’ Classifier .................................................................................... 14
2.5 A graphical model of a Markov chain ..................................................................................................... 15
3.1 $B$ versus $S$ ................................................................................................................................. 32
3.2 $\tilde{B}$ versus $S$ ........................................................................................................................... 32
3.3 Steps for Peer Migration ...................................................................................................................... 34
3.4 Finding optimum $\mu$ (Average hop count) ...................................................................................... 40
3.5 Finding optimum $\mu$ (Average resolution time) .................................................................................. 40
3.6 Effect of transfer-rate (Average hop count) ...................................................................................... 41
3.7 Effect of transfer-rate (Average resolution time) .............................................................................. 42
3.8 Convergence time for static network parameters ............................................................................. 42
3.9 Convergence time after super-peer churn .......................................................................................... 43
3.10 Social welfare cost during and after super-peer churn .................................................................... 44
4.1 Comparison of actual and estimated parameters. .................................................................................. 67
4.2 Histogram of worker accuracies for high variance regime. ............................................................... 67
4.3 Histogram of worker accuracies for low variance regime. ................................................................. 67
4.4 Pen Digits dataset: Histogram of worker accuracies with a skewed distribution of skills .................. 73
4.5 Pen Digits dataset: Histogram of worker accuracies with a more gradual “spread” of skill distribution .......................................................... 73
4.6 Dermatology dataset: Histogram of worker accuracies. Supervision greatly helps ......................... 74
4.7 Dermatology dataset: Histogram of worker accuracies. Supervision does not help much. ............. 74
4.8 Vowel dataset: Histogram of worker accuracies. ............................................................................. 76
4.9 The MTurk experiment: a few of the sample images ......................................................................... 77
4.10 Histogram of worker accuracies for the MTurk experiment. 77
5.1 A sparse cut 84
5.2 Hierarchical sparse cut detection algorithm: Starting with randomized clustering, inter cluster sparse cut detection, partition collapse, incrementing the hierarchy and randomized clustering at the next hierarchy. 89
5.3 The effect of number of attack edges on detection probability. The number of legitimate sybil nodes is constant at 2. 93
5.4 The effect of number of legitimate sybil nodes on the detection probability. The number of attack edges is constant at 4. 94
5.5 False positives for different sparsity threshold values on graphs with varying small-world property. 94
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Comparison of two cost frameworks ($C_0$ and $\tilde{C}_0$)</td>
<td>30</td>
</tr>
<tr>
<td>4.1</td>
<td>Synthetic data generated using the stochastic model for the high variance regime: Changing task difficulty variance</td>
<td>68</td>
</tr>
<tr>
<td>4.2</td>
<td>Synthetic data generated using the stochastic model for the low variance regime: Changing proportion of spammers</td>
<td>68</td>
</tr>
<tr>
<td>4.3</td>
<td>Synthetic data generated using the stochastic model for the high variance regime: Changing number of worker assignments</td>
<td>69</td>
</tr>
<tr>
<td>4.4</td>
<td>Synthetic data: Comparing objective-based methods for spammer, hammer and adversary model</td>
<td>69</td>
</tr>
<tr>
<td>4.5</td>
<td>Synthetic data: Comparing objective-based methods for data generated using model in Section 4.2.2</td>
<td>70</td>
</tr>
<tr>
<td>4.6</td>
<td>Comparing US-SW and US-HYB</td>
<td>70</td>
</tr>
<tr>
<td>4.7</td>
<td>Experiments using Pen Digits dataset: Average number of erroneous tasks</td>
<td>72</td>
</tr>
<tr>
<td>4.8</td>
<td>Experiments using Dermatology dataset: Average number of erroneous tasks</td>
<td>75</td>
</tr>
<tr>
<td>4.9</td>
<td>Experiments using Vowel dataset: Average number of erroneous tasks</td>
<td>75</td>
</tr>
<tr>
<td>4.10</td>
<td>Experiments using Nominal dataset: Average number of erroneous tasks</td>
<td>76</td>
</tr>
<tr>
<td>4.11</td>
<td>MTurk Experiment: Average number of erroneous tasks</td>
<td>78</td>
</tr>
</tbody>
</table>
Acknowledgments

Four years ago I came to Penn State University with a hope of making a difference in the research area of my interest. Just when I was wondering whether I could fit into the scheme of things, my advisor Dr George Kesidis believed in me. Since then, he has been more than just an advisor, in fact he has been a coach during my graduate life. Thank you, Dr Kesidis for being a terrific guide. You were the silent driving force and were able to extract the best from me. I cherish all the one-on-one meetings that gave this newcomer a much needed perspective and encouragement towards holistic development of career and abilities, even in those areas that did not directly relate to my thesis. Without the space and freedom you have given me, I may have burnt out long time back. I am grateful that you believed in my ideas and carefully guided me to make them into a consummate thesis material.

I am also grateful to Dr David Miller, whose meticulous and detail-oriented approach will help me long way in life. I have learnt from him the art of perfecting at the most minute level of detail which is a trademark of high quality work. His keen interest in my research has always given me an extra push and introduced to me areas that were not in my comfort zone.

I would also like to thank Dr Christopher Griffin for laying my foundations in game theory and also guiding in the area of incentive design in P2P systems. I would also like to thank Dr John Doherty and Dr Jason Morton for their invaluable guidance and support over the years. Our interactions have given me much needed learning experience that continues to educate and enrich my life.

My heartfelt gratitude to my parents for being a pillar of support all these years. I wish that I can someday do for my kids at least 10 per cent of all that they have done for me. And dad, for being proud of me enough to tell everyone (when Im not listening, of course) about even the smallest of my achievements since childhood. I know youll never read this and thats the only reason I am able to say this.

I am indebted to Sri Sri Radha Damodar, H.H. Ramapada Swami and Gita Na-
gari Yoga Farm family, especially Dhruva Maharaja Prabhu and Parijata Mataji. All that I know about life, I learnt on the farm.

I would like to express my heartfelt gratitude to Narayanan and Vidhya, Chandra and Smitha, Shrawan and Shilpa, Rahul and Supriya, Shiva, Nausheel, Arnab, Ravi, Vamshi, Chandraprakash, Rahul and Penn State Vedic society members for those endless hours of fun, singing, potluck dinners, laughter and debates. I will surely miss these days.

Finally I would like to thank my wife who has been my constant companion throughout the highs and lows of my academic life.
Dedication

Dedicated to Amma and Papa for motivating me to pursue Doctorate degree despite economic difficulties and to my wife who has been my co-pilot and partner in crime.

And

To the most beautiful lotus eyed one, whose effulgent bodily complexion defeats the luster of dark monsoon clouds. The most merciful Sri Sri Radha Damodar, bedecked with brilliant jewels and a peacock feather, who travelled all across USA in buses, spreading the sankirtana movement and attracting the hearts of the conditioned souls.
- Krsna Janmashtami, August 28, 2013
Introduction to Multi-agent Systems

Multi-agent systems consist of multiple, typically, a large number of computing units called agents that interact with each other through a common “environment”. These are autonomous, in the sense that they are capable of “deciding for themselves” the actions that satisfy their own design purpose. Agents can be used to represent a large variety of entities such as humans, robots or even parallel software units or threads. The study of multi-agent systems encompasses a variety of applications from modeling real world phenomenon, e.g., flocking of birds to solving complex software or computational problems. It is a branch of distributed artificial intelligence that has substantially evolved and diversified since its conception to include ideas from multiple disciplines such as game theory, economics, cognitive sciences and others. According to [1], there have been two motivating factors behind the rise multi-agent systems: 1) The exponential growth of computational systems having distributed, large, open and heterogeneous sub-components such as peer-to-peer (P2P) systems that offer a scalable alternative to content distribution via the traditional client-server systems. 2) The need for developing and analyzing theories of interactivities and the resulting macroscopic phenomena in societies of humans, or in a group of robots or sensors. This chapter covers a brief background on multi-agent systems, their motivation and application along with some of the related concepts in game theory and artificial intelligence. This discussion will “set the scene” for the rest of the thesis. In this chapter, we will focus only on the mathematical domain-independent concepts and the domain specific background will be discussed at the beginning of each of the following chapters.
1.1 Intelligent Agents

[1] defines an agent to be an autonomous, computational entity that perceives and interacts via a common environment which is shared with other agents. The autonomy of an agent is due to the fact that he is motivated to act using his own “discretion” based on his local parameters and his limited knowledge which could be different from that of other agents in the system. Agents are intelligent because their decisions are based on efficiently achieving their purpose or goals. Moreover, agents are interactive because their decisions are affected by the decisions of other agents. According to [2], an intelligent agent typically has the following three qualities: reactivity (to respond to changes in the environment in a timely manner), proactiveness (goal-directed behavior) and social ability (interactions with other agents).

1.2 Challenges in Multi-agent Systems

[1] distinguishes multi-agent systems into two categories: one in which several agents coordinate their knowledge and activities and reason about the process of coordination; and the second one which is a distributed problem solving system where the problem is divided among a number of nodes. The elementary question posed by multi-agent systems is “When and how should which agents interact (cooperate or compete) to successfully meet their design goals?”. To address this issue, one of two different approaches can be taken: a bottom up approach where specific agent level properties are searched in order to have some desired interaction at the over all system level; or a top-down approach where we come up with system level rules that define the interactions of the agents at the microscopic level. In chapter 3, for instance, we will study and compare both the approaches by applying it to our problem of interest in P2P systems. Some of the typical challenges in the design of multi-agent systems are discussed below:

- Task decomposition and aggregation: This forms a crucial component in the design of multi-agent systems that use a top-to-down agent-based modeling approach. One has to efficiently “map” the system goals to the local strategies that each agent employs with due consideration given to the purpose
and motivation of agents and their limited visibility.

- **Agent-level communication**: The communication protocols used by agents define the system overload and hence the efficiency.

- **Learning**: Allowing agents to represent and reason about actions, plans and knowledge of other agents.

- **Synchronization and deadlocks**: Enabling agents to recognize and reconcile disparate viewpoints and conflicts.

- **Balancing local computation and communication**.

- **Designing systems that lead to steady state equilibrium to avoid or mitigate chaotic or oscillatory behavior**.

### 1.3 Agent Architectures

The representation of agents and their dynamics is largely dependent on the domain characteristics. However their architectures can broadly be classified into two types: deterministic and probabilistic. Deterministic view of agents allows us to predict the exact decisions of agents and the state transitions of the environment. Each agent has a set of possible actions to choose from and the environment consists of a set of possible states. The set of actions of all agents together decide the transition of the environment states. In a more generalized setting, the system and agent variables depend on the history of states and actions, i.e., a system with memory. However a class of agents called purely reactive agents decide what to do without reference to their history. In chapter 3, we will study such purely reactive agents (peers) in a P2P setting. The probabilistic view of agents allows for a formal study in the absence of complete information. In such a setting, agents can be characterized by random parameters. For instance, in a scenario where agents can be either honest or malicious or the environment can be one of many known types, a single random variable may be used to represent agent’s intention or the environment type. In chapter 5, we utilize machine learning tools to account for missing information and “learn” unknown agent parameters.
1.4 Agent Utility

The actions of all agents are purpose-oriented, i.e., they are executed with an intention towards achieving specific “selfish” goals. Utility functions help us model agent behavior for a certain environment state or a history of environment states. Every agent $i$ is associated with a utility function $u_i : E \rightarrow \mathbb{R}$ that maps the current state of the environment $E$ to a real value. Note that the state of the environment is typically decided by the history of actions of all the agents (in a system with memory) or simply the current choice of action of all agents (in a memoryless system). The utility functions characterize both selfish as well as interactive nature of multi-agent systems: selfish, because each agent tries to maximize its utility function (or minimize its cost function) and interactive, because each agent’s utility depends not only on his own actions but also on the actions of other agents. Not surprisingly, in such a setting game theoretic tools are naturally suitable and hence used often in their analysis [3].

1.5 Multi-agent Interactions

Game theoretic tools such as auction theory, Nash equilibria offer a powerful means of studying the interactive multi-agent dynamics. Dependence relations between agents decide the exact mathematical tools used for their study. For instance, market and pricing dynamics with agents being the customers, retailers or suppliers are well characterized by auction theory. Co-operative and competitive Nash equilibrium concepts can be used to find the possibility whether a fixed stable point exists in the system. In this regard, it is necessary to define a Nash equilibrium.

1.5.1 Nash Equilibrium

Informally, a Nash equilibrium is a “stalemate” state of agent actions, where none of the agents can increase his utility (or reduce his cost) by unilaterally changing his action. Hence, at Nash equilibrium, none of the agents are motivated to change their choice of actions unilaterally (not by “collusion”). Consider a system of $n$ agents where $A_i$ is the set of all possible actions for agent $i$. We denote the set of action profiles of all agents as $A = A_1 \times A_2 \times ... \times A_n$. Let $u_i : A \rightarrow \mathbb{R}$ be the
utility function of agent \( i \). Let \( a_i \in A_i \) be the choice of action of agent \( i \) and \( a_{-i} \) denote the action profiles of all agents except \( i \). An action profile \( a^* \in A \) is a Nash equilibrium if no unilateral deviation in strategy by any single agent is profitable for that agent, i.e.,

\[
\forall i, \ a_i \in A_i, \ u_i(a_i^*, a_{-i}^*) \geq u_i(a_i, a_{-i}^*)
\]  
(1.1)

### 1.5.2 Convergence to Nash Equilibria and Potential Games

Note that the existence of a Nash Equilibrium does not guarantee that the system of agents will actually reach a point of stability. In order for the system to actually settle down at a Nash Equilibrium, one has to study the convergence properties. In particular, the local dynamics at each agent guarantee stability if we can find an associated “Potential” or a “Lyapunov” function. Such functions are widely used to prove stability of both linear and non-linear control systems.

A game is said to be a potential game if the incentive of all players to change their strategy can be expressed using a single global function called the potential function. The potential function is a useful tool to analyze equilibrium properties of games, since the incentives of all players are mapped into one function, and the set of pure Nash equilibria can be found by locating the local optima of the potential function. Convergence and finite-time convergence of an iterated game towards a Nash equilibrium can also be understood by studying the potential function. A generalized potential game is one where there exists a function \( \Phi : A \rightarrow \mathbb{R} \) such that \( \forall a_{-i} \in A_{-i}, \forall a'_i, a''_i \in A_i, \)

\[
u_i(a'_i, a_{-i}) - u_i(a''_i, a_{-i}) > 0 \Rightarrow \Phi(a'_i, a_{-i}) - \Phi(a''_i, a_{-i}) > 0
\]  
(1.2)

A weighted potential game satisfies a stronger condition:

\[
\Phi(a'_i, a_{-i}) - \Phi(a''_i, a_{-i}) = w_i(u_i(a'_i, a_{-i}) - u_i(a''_i, a_{-i}))
\]  
(1.3)

for some constant \( w_i \in \mathbb{R} \). For a potential game, we can easily see that iterative improvement in local choice of an agent’s action will guarantee an improvement in the potential function. In the case where the potential function is bounded above,
we can prove that the system will converge to a Nash equilibrium point.

1.6 Agent-based Modeling

The paradigm of agent-based modeling uses the bottom-up approach to characterize, represent, predict or recreate a system or a phenomenon. The system is characterized or described in terms of the actions, behavior, beliefs, etc. of an individual autonomous agent in a group of large number of such autonomous agents. A common underlying theme motivating agent based modeling is that most complex phenomena observed in the physical world are the consequence of a much simpler set of rules that govern the dynamics of a large number of constituent entities that it is comprised of. For example, the Axelrod’s model of social dissemination [4] tries to explain the consensus towards cultures and the simultaneous existence of different cultures in the society using a simple set of rules of interaction between individuals or behavioral models that explain the flocking of birds [5] or ant colony optimization [6]. Agent-based modeling has received a large amount of interest in the last decade even from the non-computing research communities in areas such as social sciences and ecology [7]. It is a comparatively new method of modeling and a large number of existing models can be extended under this paradigm. [8] illustrates how the classic predator-prey model can be enriched by an agent-based model that makes more realistic assumptions without any significant addition to the model complexity. A large number of agent-based modeling toolkits such as NetLogo [9] and Repast [10] have been designed in the last decade. A useful comparative evaluation of the existing toolkits has been provided in [11].

1.7 Multi-agent Learning

Classical game theoretic models are based on the assumption that the rationality and the preferences (utilities) of the agents are a common knowledge. However, this is not always a realistic assumption. In most cases, the equilibria observed in real world systems are a result of repetitive game play between the agents, where they gradually learn about the rationality and preferences of other agents and subsequently adapt their strategies based on the incremental information gain after each
game play. This is one of the many ways in which learning can be closely associated with the study of multi-agent systems. For instance, [12] studies the introduction of learning to multi-agent systems to account for the bounded rationality of the agents and the incomplete local information at each agent. In some cases, the bounded rationality argument translates into accounting for heterogeneity in the agent’s abilities to choose the best possible strategy. For instance, in Chapter 4, we will study how agents choose their responses from a categorical strategy space based on agent-specific parametric values which are a priori unknown.

An important aspect of multi-agent systems is dealing with the behavior management in collections of several independent entities, or agents [13]. In fact, multi-agent systems is considered a subfield of artificial intelligence (AI) that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents behaviors. [13] considers two important aspects of multi-agent systems from the perspective of learning: degree of heterogeneity and degree of communication. Based on these aspects, one can consider four different scenarios: homogeneous non-communicating agents, heterogeneous non-communicating agents, homogeneous communicating agents, and heterogeneous communicating agents. We will focus now on the heterogeneous and non-communicating scenario which describes most of the problem cases studied as a part of this thesis. Apart from the fact that agents are situated differently in the environment which causes them to have different sensory inputs and necessitates their taking different actions, the agents have much more significant differences. They may have different goals, actions, and/or domain knowledge. Multi-agent systems applied to the study of robust security policies often consider two broad categories of agents: honest and malicious. Malicious agents are attackers that leverage the loopholes in the system to their advantage. Note that in such a case a learning mechanism needs to be employed by the agents or the system as a whole to detect and neutralize the attackers. Often, in such cases, one has to infer the intention of the agent based on the agent’s observed behavior. This notion can be formalized as learning the unknown parameters based on some observed data. This is a classical machine learning problem and hence, we can exploit machine learning principles. In the following we will discuss briefly some important machine learning concepts that are related to this thesis.
1.8 Summary of Research Contributions

The major contributions of this thesis are as follows:

**A local cost function for peers to choose their super-peers based on a novel game-theoretic task decomposition framework:**

Efficient resolution of a query in a super-peer based P2P system relies on the quality of clusters formed by the peers. Various local figures of merit can be used by a peer while choosing its super-peer. Our contributions in here are:

- Mathematically modeling the average query resolution time considering the semantic similarity of peers within each cluster along with the load at each cluster.

- Design of an incentive mechanism for the peers and a distributed cluster optimization algorithm that is guaranteed to converge if we assume that the network characteristics (peer relationships and super-peer parameters) change gradually, as compared with the convergence time of the algorithm.

**Novel task aggregation approaches in crowdsourcing: Accounting for heterogeneity in agent skill, intention and task difficulty:**

The inference of ground truth answers in a crowdsourcing system such as Amazon MTurk [14] relies on the rules to aggregate the answers provided by a crowd of workers. Our main contributions in this area are:

- An intuitive generative modeling approach based on the difference between the task difficulty and the worker’s skill level.

- A logical and mathematical dissection of worker reliability into skill level and intention.

- Extension of the concept of adversarial workers to multicategory annotation tasks and the introduction of more strategic and complex adversarial agent behavior.

- A novel weight energy constrained approach that maximizes the aggregate confidence of the ground truth answers.
• A novel “black-box” method of experimental analysis of crowd-aggregation scheme using an ensemble of weak classifiers to represent a worker (agent) in the crowd.

**A distributed hierarchical sybil detection algorithm:**
Distributed reputation systems can be employed in multi-agent systems to enforce or at least encourage honest behavior. However, the reputation scores can be easily manipulated by sybil identities, *i.e.*, (multiple) identities that are “owned” by a single malicious agent. Our contributions in this area are:

• A distributed graph theoretic algorithm on trust graphs and a sparsity metric that indicates presence of sybils.

• The algorithm uses hierarchical methods, and at each hierarchy the information is aggregated and condensed using a novel partition collapse method, thus reducing the overall communication and computation cost.
Background on Machine Learning

2.1 Supervised, Unsupervised and Semi-supervised Learning

In a broad sense, machine learning involves obtaining an algorithm that takes input as a feature set and maps it to the output set or the set of predictions. If the output set is continuous then, such a classification is referred to as regression, whereas if the output set is categorical then, it is referred to as classification. Note that the input or the domain of the function can be either discrete, continuous or mixed. In the case of supervised learning, one is provided with a labeled data set, i.e., a set of input features for which the output known as the “ground truth” is known. Given a set of training examples of the form \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \), a supervised learning algorithm seeks a function \( g : X \rightarrow Y \), where \( X \) is the input space and \( Y \) is the output space. The function \( g \) is an element of some space of possible functions \( G \), usually called the hypothesis space. It is sometimes convenient to represent \( g \) using a scoring function \( f : X \times Y \rightarrow \mathbb{R} \) such that \( g \) is defined as returning the \( y \) value that gives the highest score: \( g(x) = \arg \max_y f(x, y) \). Let \( F \) denote the space of scoring functions. Although \( G \) and \( F \) can be any space of functions, many learning algorithms are probabilistic models where \( g \) takes the form of a conditional probability model \( g(x) = P(y|x) \), or \( f \) takes the form of a joint probability model \( f(x, y) = P(x, y) \). For example, naive Bayes’ and linear discriminant analysis are joint probability models, whereas logistic regression is a conditional probability
model. In the case of unsupervised, learning one does not have the luxury of a labeled data set. Unsupervised learning relies on finding hidden structure in the unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution. This distinguishes unsupervised learning from supervised learning. Approaches using unsupervised learning mostly use clustering methods such as K-means or Gaussian Mixture Models. Semi-supervised learning is a class of machine learning techniques that make use of both labeled and unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised and supervised learning. It has been generally observed that unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement over purely supervised approaches in the learning accuracy. The acquisition of labeled data for a learning problem comes at a cost. In such situations, semi-supervised learning can be of great practical value. Semi-supervised learning is also of theoretical interest in machine learning and as a model for human learning. Semi-supervised learning may refer to either transductive learning or inductive learning. The goal of transductive learning is to infer the correct labels for the given unlabeled data only. The goal of inductive learning is to infer the correct mapping from X to Y which can be directly applied in future to a new set of unlabeled test data.

2.2 Generative and Discriminative Classification

Discriminative classifiers directly learn the feature conditional probability of the output \( P(y|x) \) from the training data without modeling the probability distribution of all the variables (including the latent variables). Generative models infer the distribution over all the variables in the form of a joint distribution or a pmf \( P(x, y) \). Note that joint distribution contains more information than a conditional distribution and all the marginal as well as conditional distributions can be easily derived from a joint distribution. So, essentially, discriminative models simply learn a boundary or a criterion for separating the feature space into the categories, whereas a generative model gives a complete expression of how the features are stochastically generated. Discriminative classification generally involves
less computation and allows us to use several off the shelf tools such as support vector machines (SVMs), logistic regression methods including kernel-based techniques, artificial neural networks and so on. Figure 2.1 shows a boundary over a two dimensional feature space separating two class decision spaces.

Generative models are often employed in unsupervised and semi-supervised methods since the absence or dearth of labeled data necessitates the expression of the underlying patterns in the unlabeled data using a stochastic generation model. The joint and class conditional distributions are computed from the assumed data generation model. A very commonly used generative probabilistic model is a Gaussian mixture model (GMM) in which the data is generated by making a “soft” decision between a set of Gaussian distributions, each with different parameters (mean and variance). Figure 2.2 shows data labeled according to a generative classifier employing a GMM.
2.3 Generative Probabilistic Modeling

In generative modeling based classification, one makes assumptions about the underlying joint probability distribution. This distribution can be characterized by unknown parameters which act as additional degrees of freedom while fitting the model to the data. A Naive Bayes’ classifier is a generative classifier based on the assumption that each feature is conditionally independent of the others given the true class that is used to generate the features. Note that this assumption simplifies numerical computation and, although seemingly unrealistic, it has been widely used in topic based document clustering under the “bag-of-words” model. In many scenarios, the conditional dependence of one feature on the others cannot be ignored. A graphical model and a Bayes’ network in particular can be used to succinctly represent conditional dependencies and independence between the random variables. Each node in the graphical model represents a random variable and each edge represents dependency. For a Bayes’ network, which is an acyclic graph, the joint distribution over all the variables is given by:

\[ P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | pa(X_i)) \]  

(2.1)
Figure 2.3: A graphical model of health diagnosis for predicting diseases

Figure 2.4: A graphical model of a Naive Bayes’ Classifier

where $pa(X_i)$ is the immediate parent of $X_i$ in the graph.

In figure 2.3, a graphical model used to represent the dependencies over variables used for predicting diseases. From the model, it is assumed that the occurrence of lung cancer is independent of bronchitis given that the patient is addicted to smoking. We can also categorize the naive Bayes’ classifier and the Markov chain as a simplistic examples of graphical models as shown in figures 2.4 and 2.5.

When all the variables in a graphical model are discrete, then the value of their pmfs for each realization of the random variables form the parameters of the
model that need to be estimated. In the case of continuous random variables, the
parameters of the distribution such as mean, variance and so on can be used as
the parameters of the model. For instance, Gaussian mixture models (GMMs) are
widely used in data clustering where the data (a set feature vectors with a distance
metric) is hypothesized to be generated by first choosing a Gaussian model from
a set of Gaussian models and then using that model to generate the data. Each
Gaussian model is characterized by different set of parameters (mean and variance).

2.4 Incomplete Data and EM algorithm

When the training data consists of information of all the variables in the model,
then it is considered to be complete. For instance, if all the variables in figure 2.3,
are observed in the training data, then our job is to simply compute the param-
eters via maximum likelihood estimation, i.e., we maximize the joint probability
over all the variables of observing the training data. This trained model with fixed
parameters is applied to the test data to predict results and evaluate performance.
Note that in some cases, not all the variables can be observed, for instance, in
clustering problems using GMM, we do not know the cluster assignment variable
or the hidden states in a hidden Markov model used for speech recognition. In
such cases we are not able to directly compute the likelihood of the training data.
The Expectation Maximization algorithm first proposed by Dempster, Laird and
Rubin [15], is a popular tool used in such scenarios. It is an iterative method,
where, each iteration consists of two step: Expectation (E) step and Maximization
(M) step. In the E step, the missing data is estimated based on the observed data
and the current estimate of the unknown model parameters. In the M-step, the
likelihood of the observed data is maximized based on the information (distri-
bution) of the missing data. We can guarantee convergence because at each iteration,
the likelihood function increases in value.
Suppose that the observed set of variables is denoted by $\mathcal{X}$, the unobserved variable set is denoted by $\mathcal{Y}$ and the set of model parameters by $\Theta$. We can express the complete data as the union of complete and incomplete data $\mathcal{Z} = (\mathcal{X}; \mathcal{Y})$. In a non-bayesian approach to data modeling, we consider the parameters of the model to be deterministic, so we can express the log likelihood as a function of the parameters. In a Bayesian approach, one needs to also consider the distribution over the model parameters and take their priors into account. $\mathcal{L}(\mathcal{X}, \mathcal{Y}|\Theta)$ and $\mathcal{L}(\mathcal{X}|\Theta)$ represents the complete and incomplete log-likelihoods respectively.

Suppose we know the distribution over the unobserved variable $\mathcal{Y}$ conditioned on the knowledge of $\Theta$, then we can compute the expected complete log-likelihood as $E[\mathcal{L}(\mathcal{X}, \mathcal{Y}|\Theta)]$. In the M-step, we maximize this value over all possible values of $\Theta$.

$$\Theta^* = \arg \max_{\Theta} E[\mathcal{L}(\mathcal{X}, \mathcal{Y}|\Theta)]$$ \hspace{1cm} (2.2)

In the E-step, we recompute the distribution of $\mathcal{Y}$ based on the value of $\Theta$ computed in the previous M-step.

$$P(\mathcal{Y}|\mathcal{X}, \Theta)$$ \hspace{1cm} (2.3)

We can prove that the E and M steps descend in the incomplete log-likelihood and converge to a locally optimal point.
Chapter 3

Efficient Query Resolution by Optimization of Cluster Formation in Super-Peer Networks

3.1 Introduction

The rapid growth of peer-to-peer (P2P) systems, due to their inherent scalable architecture that supports massive content distribution among the users/peers, has led to an aggressive effort to optimize their performance. P2P based applications [16] [17] [18] account for a large portion of today’s Internet traffic, with potential for further growth. Distributed mechanisms for storing and addressing content, fair distribution of load, and robustness against peer churn are some of the well-known issues associated with P2P systems. As the network size grows, so does the synchronization overhead. In the case of unstructured P2P networks, the overhead of query forwarding and of storage and update of caches is even more severe. To address this, a hierarchical approach based on leveraging the heterogeneity of the participating peers can be used. The idea of super-peer based networks [19] designates a few peers in the overlay network as super-peers to act on behalf of the peers assigned to them. The super-peers have resource (memory, computation and bandwidth) capacities much greater than other peers. Ordinary peers (henceforth just referred to as peers) are assigned to at least one super-peer and
they route all their queries through this super-peer. The super-peers themselves form a sub-overlay as a regular P2P network. Super-peers tend to become hubs of computation and communication as they are directly involved in the content distribution protocol for all the peers dependent on them. This asymmetric model has the combined advantages of semi-centralized search as well as a scalable distributed system. While the above gives the idea of a super-peer based network, the implementation-specific details related to query handling, data and pointer storage in the cache, and the type of overlay may vary for each model.

In this work we focus on a super-peer based architecture. Particularly, we consider a model wherein peers generate queries that may have semantic similarity with the queries of other peers [20]. In this way, one should be able to classify the queries by their semantic types. One can thus suggest mechanisms to allow self organization of super-peer based systems, with peers with similar “semantic” interests encouraged to choose the same super-peer. In such a case, a query can be resolved within the cluster assigned to the super-peer with high probability without the need for any inter-cluster query forwarding or processing. However, identifying and classifying nodes and queries based on their semantic types (or communities of interest) may be difficult, especially in the distributed and ad-hoc setting where the semantic proximity measures might vary from peer to peer. In our model, we allow peers to learn about their relationship with other peers based on the history of transactions. The inferred relationships may change as the interests of peers potentially evolve over time. Our model is flexible enough to incorporate these dynamics of peer behaviors and propensities.

We also consider the transactional load generated by each peer in terms of the average rate of queries generated at its super-peer. Hence the peer to super-peer assignment considers both semantic similarity as well as the computational load at each super-peer. This model can be fully expressed as a graph of ordinary peers wherein nodes represent the peers, their (node) weight represents their query rate, and the edge weights represent the semantic proximity between the peers. This graph has to be partitioned between the super-peers to achieve an overall system goal. In our simulation study, we focus on the average query resolution time as the system’s figure of merit.

Our main contribution is the design of an incentive mechanism for the peers and
a distributed cluster optimization algorithm that is guaranteed to converge if we assume that the network characteristics (peer relationships and super-peer parameters) change gradually, as compared with the convergence time of the algorithm. If we assume that each player (peer) selfishly decides to choose its super-peer based on the proposed criteria, then the system of peers drifts towards a clustering assignment optimum of an overall cost function that reflects the "social welfare" of the system [21]. This local cost approximates the average query resolution time. We evaluate the accuracy of this approximation in our experiments. We also prove that convergence to such locally optimal points is guaranteed. The incentive can be calculated entirely based on the local information easily accessible to a peer and from the peer relationship information inferred from the transaction history. The incentive is also intuitive in the sense that it encourages a peer to connect to a super-peer that will process its queries most efficiently.

Based on the local cost criterion, each peer selfishly decides its super-peer and this gives rise to a greedy algorithm where peers can asynchronously decide to switch between super-peers, allowing for a self-adaptive system. We show that this algorithm descends in a global cost that reflects the load variance across, and the semantic disparity within, the super-peer clusters. We show empirically that the local cost based approach performs closely with respect to a similar iterative scheme that the uses direct minimization of a centralized cost criterion. In fact, for certain values of the relative weight parameter (described in the following sections) the former performs better. We also intuitively explain this performance advantage. Moreover, the local (peer-level) decomposition based on the centralized criterion requires use of additional shared variables adding to the synchronization overhead. Finally, we demonstrate the algorithm on a discrete-time simulator to evaluate the proposed scheme. We study the performance of the system based on the time for query resolution and robustness to peer churn.

The remainder of this chapter is organized as follows: In Section 2, we describe the background and relevant literature. In Section 3, we describe the system architecture and motivate the need to have local decision criteria. In Section 4, we formalize our problem and introduce a peer-level cost function, formulate our game, and prove the existence of Nash equilibria in pure strategies. Section 4 also explains the proposed algorithm based on the cost framework. Some methods to
improve the locally optimal equilibrium points are also discussed in this section. In Section 5, we introduce a centralized version of the problem and a comparative numerical study between the two frameworks. In Section 6, we describe the details of the local incentive based algorithm that leverages the game-theoretic payoff function. In Section 7, we describe the simulator model and present our empirical findings in Section 8. Section 9 concludes the discussion with pointers to possible future work.

3.2 Background

For over a decade, P2P based applications have provided an alternative framework to the conventional client-server model, wherein the application nodes known as peers share the role of both client and server and thus, with each peer brings its own set of resources that can be potentially used by others in the system. The peers are connected to one or more peers running the P2P application thus forming an overlay network, distinguishing it from the physical network connecting end-hosts and routers in the Internet. Peers interact with each other on this overlay network. The interaction could be a query search, a peer search, range search, etc. The decentralized model of content storage and distribution comes with a set of challenges such as the need to store, address and retrieve the data from peers that may have intermittent life spans. In structured networks [22] using algorithms like Chord [23] or CAN [24], the topology is regular and nodes are connected and assigned responsibility for the content based on a Distributed Hash Table (DHT). These networks employ a well-defined query search with performance guarantees. Unstructured networks [17], on the other hand, allow for arbitrary peer connections and queries are usually limited-scope flooded through the network until a match is found. Our model of the P2P file sharing system that we used in our simulations is based on a Gnutella-like [17] model which considers unstructured overlay, with query resolution done via random forwarding.

An issue with P2P networks is that they often assume the capabilities of the participating peers are symmetric. However studies such as [25] have shown that there exist heterogeneities in the system that give rise to bottlenecks, thus affecting the system performance. A super-peer based P2P architecture has at least two
hierarchical levels, \textit{i.e.}, the super-peers and the ordinary peers. Super-peers are nodes with resource capabilities much higher than those of ordinary peers. Accordingly, they assume higher responsibilities. Ordinary peers route their queries via the super-peers they are assigned to. Super-peers can also decide to store the pointers to the data “owned” by its peers which completely eliminates the role of the ordinary peers in processing and forwarding queries, \textit{i.e.}, an index server system like KaZaa [18]. The overlay network consisting of just the super-peers can be either structured or unstructured, with the possibility of having different query routing protocols. Edutella [26] and Skype [27] are examples of super-peer based networks.

Ideally, one would like to localize the query searches within a subset of super-peers that would give a query hit with high probability. To achieve this, it is desirable to group peers with similar content and interests together. The advantages of this are two-fold. Firstly, we can classify groups semantically, thus localizing the query to a certain group of super-peers based on the query type. Secondly, the peers will be able to solve their queries within the super-peer cluster with high probability. [28] proposes a self-organizing network to increase content locality with the semantic closeness estimated by an affinity matrix. [29] proposes a hierarchical algorithm wherein peers form clusters in a decentralized manner based on proximity in a fixed dimensional feature space partitioned by semantic type. A more comprehensive survey of distributed formation of semantic overlay networks (SONs) is given in [30]. Clustering based on measures of Resource Description Framework (RDF) schema-based classification have been proposed in [31]. These schemes are not easy to implement in the distributed ad-hoc setting of P2P systems. Some of the more feasible schemes such as “interest-based locality” [32] are based on inferring peer interest proximity based on transactional outcomes. The inference about semantic proximity could also be based on the cache over-lap of the peers [33] along with other parameters such as peer generosity and file popularity [34]. In this chapter we leverage the idea in, e.g., [32] to allow peers to learn about other peers from the past history of transactions. The intuitive idea is that a requested peer is more likely to satisfy future queries if it has been successful in satisfying similar queries in the past. Each peer maintains a proximity table of peers with whom it interacted in the past and the count of queries satisfied by
each peer. The instantaneous aggregate information present at the peers about
the semantic relationships and the query rate can be represented by a weighted
graph. The weight of each node (peer) is the rate at which it generates queries.
The edge weights correspond to the weights in the proximity table.

In this work we propose a self-organizing network of peers that choose their
super-peers based on two criteria: their semantic proximity with other peers cur-
rently assigned to the same super-peer and the existing computation burden on
the super-peer. We express this as a greedy incentive based decision that also
optimizes a global cost. The global cost reflects the instantaneous load disparity
across the super-peers along with semantic proximity of peers within super-peer
cluster. One can also express this as a graph partitioning problem of the graph
described above. [28] considers a similar framework in which peers can choose
to be part of a cluster based on semantic proximity. Similarly, in [35] the peers,
choose the super-peer offering the best search performance based on the history
of transactions. Unlike [28], [35], we propose a game-theoretic formulation of the
peer assignment where the decisions of the peers are based on the proposed in-
centive. We are particularly interested in the class of games known as “potential
games” [21] which guarantee a descent in a global cost function (indicative of the
system performance) with each decision made at the local (peer) level. Note that
load balancing has been explored in many works, most prominently in [36] for
However, the load balancing mechanism in [35] is realized by allowing overloaded
super-peers to deliberately restrict queries from some of their peers. [37] considers
the problem of locating super-peers within a “latency-based” metric space that
will load-balance the network of super-peers.

The graph partitioning problem is described as follows: Let $G = (V, E)$ be an
undirected graph where $V$ is the set of peers and $E$ is the set of edges. Suppose
the nodes and edges are weighted. Let $w_i$ represent the weight of the $i$th node and
let $c_{ij}$ represent the weight of the undirected edge $\{i, j\}$. Then the $K$-way graph
partitioning problem aims to find $K$ subsets $V_1, V_2, ..., V_K$ such that $V_i \cap V_j = \emptyset$
$\forall i, j$ and $\bigcup_{i=1}^{K} V_i = V, \sum_{j \in V_i} w_j = \frac{\sum_{k=1}^{K} w_k}{K} \forall j$ and with the sum of the weights of
edges whose incident vertices belong to different subsets minimized. It is known
to be an NP-hard problem [38].
Heuristics to solve the graph partitioning problem primarily make use of spectral bisection methods [39] or multilevel coarsening and refinement techniques [40]. Spectral bisection methods calculate the eigenvector corresponding to the second smallest eigenvalue, known as the Fiedler vector, of the modified adjacency matrix of the graph. These methods by far give the best results. However, finding the Fiedler vector is computationally very expensive. For “geometric” graphs in which coordinates are associated with the vertices, geometric methods are available which are randomized and quicker than spectral bisection methods. Multilevel partitioning algorithms are by far the most popular techniques. The idea is to coarsen the graph, i.e., locally, at different places in the graph, collapse a connected component into a single node, until the graph is reduced to a small number of such nodes that can be partitioned easily (using brute force methods). The subsequent uncoarsening stage is accompanied by refinement of the partition. The coarsening phase is improvised by using random matching edges or finding highly connected sets of nodes. The partitioning phase may use spectral bisection, the K-L method [41], or the graph growing partition algorithm (GGP) [42]. The K-L method is used for step by step refinement to identify groups of nodes that can be exchanged between two partitions in order to improve the overall partition. All the previously discussed methods are motivated by a global criterion used to define an optimal partitioning and hence are most suitable for centralized computational implementations involving access to global state information.

### 3.3 System Overview

We first begin with an overview of our model.

- Every peer is assigned to a single super-peer. Note that some methods [35] allow peers to connect to multiple super-peers at the expense of cache redundancy at multiple super-peers. Such systems are more fault tolerant to super-peer faults. However, in our self-adaptive model, the peers would reassign themselves to another super-peer in case of failure or a bottleneck. This is because each peer maintains a list of potential alternative super-peers that it can connect to as will be described later.
Super-peers might have different capacities. The normalized capacity of the $k^{th}$ super-peer in terms of the number of queries per unit time is defined as:

$$w_k = \frac{s_k}{\sum_{j=1}^{K} s_j},$$

where $s_j$ is the unnormalized capacity of the $j^{th}$ super-peer in terms of the number of queries per unit time and the total number of super-peers is $K$. The normalized capacity can be computed synchronously between the super-peers by exchanging information along with query forwarding. This is feasible since the number of super-peers is typically far less than the number of peers.

Every super-peer maintains a cache of file pointers or identifiers owned by its peers. A peer join results in the peer communicating the list of file pointers owned by it to the super-peer. This speeds up query search as the resource-poor peers are kept out of the search operation. Similarly, every peer departure results in the deletion of its cache at the super-peer.

Each peer maintains a proximity table which contains four fields: peer-id, super-peer-id, load and query-hit-cnt. After every successful query resolution, the peer updates the details of the peer that resolved the query in the proximity table. The table fields are further described below:

1. peer-id: Contains the identifier (IP address) of the peer
2. super-peer-id: Contains the identifier (IP address) of the super-peer that the peer is currently assigned to.
3. load: Contains the current effective load of the super-peer. The effective load is the load divided by normalized capacity of the super-peer.
4. query-hit-cnt: Contains the number of queries of the evaluating peers satisfied by peer-id added with the number of queries of the peer-id that are satisfied by the evaluating peer. The query-hit-cnt is representative of the semantic similarity between the two peers.

In order to facilitate the decision making mechanism, the information from the proximity table is sorted out in a super-peer table that has three fields:
super-peer id, load, and proximity. In this table, a list of super-peers known to the peer is maintained along with their corresponding loads and semantic proximity stored in proximity. The semantic proximity value is the sum of the query-hit-cnt of all the known peers that reside on the super-peer.

- Each peer computes the query rate as the average number of queries generated by itself, which are averaged over a fixed time window. The time window is a fixed globally known parameter. We denote the query rate for a peer \( i \) as \( b_i \) in our problem formulation below.

- The load at super-peer \( k \) is the sum of the query rates of all the peers that it serves.

- The super-peer that a peer \( i \) chooses is denoted \( r_i \).

### 3.4 A Local Cost Function

One way of formulating a game for the problem of super-peer assignment could be by treating super-peers as players with each super-peer bidding for the peers that help to maximize its utility or minimize its cost. This can be formulated as a combinatorial auction game. Combinatorial auctions have been studied widely [43]. On the other hand, treating the peers as players who may choose from among super-peers to minimize their own costs is an alternative approach to formulating the partitioning problem as a game. This is our approach. In the following, we specify suitable cost functions that guarantee a Nash equilibrium and give us a locally optimum partition in the sense of a global potential (Lyapunov) function, approximating total simulation time.

Suppose there are \( N \) peers and \( K \) super-peers. Let \( r_i \in \{1, 2, \ldots, K\} \) be the super-peer chosen by the \( i \)th peer. Let us assign the following cost function to the \( i \)th peer.

\[
C_i(r_i, r_{-i}) = \frac{b_i}{w_{r_i}} \sum_{j:j \neq i, j \neq r_i} b_j + \frac{\mu}{2} \sum_{j:j \neq r_i, j \neq r_i} c_{ij},
\]

(3.1)

where \( r_{-i} \) denotes the vector of assignments of all the peers except that of the \( i \)th
The first term represents the cost due to load at the super-peer that the peer is assigned to whereas the second term represents the semantic proximity of the peer with other peers that do not belong to the same super-peer. Here, $\mu$ denotes the relative weight given to this semantic proximity cost. Let us focus on the query processing cost. In the function above, this cost to a peer intuitively depends on two factors: the existing load on the assigned super-peer, i.e.,

$$
\sum_{j: r_j = r, j \neq i} b_j, \tag{3.2}
$$

and the computational load that the peer will bring to the super-peer due to the queries generated by it, i.e., $b_i$. Supposing that for a hypothetical peer, the computational load generated by the peer is zero ($b_i = 0$), then the computational part of the cost should be zero. In this way, multiplication by $b_i$ makes sense. This cost incentivizes the peer to choose a super-peer that has relatively less existing load, thus encouraging load balancing at the system level. For example, if $\mu = 0$ then a peer $i$ which is currently assigned to super-peer $r_i$ would choose to relocate to super-peer $r_i^*$ only if

$$
\frac{1}{w_{r_i}} \sum_{j: r_j = r, j \neq i} b_j > \frac{1}{w_{r_i^*}} \sum_{j: r_j = r_i^*} b_j, \tag{3.3}
$$

and in this way the load balancing mechanism is implicitly manifested in the local cost function. The second term in the sum represents the weight of edges that connect the $i$th node with nodes in other partitions. Note that the variables needed to compute this local cost are available locally at each peer.

### 3.4.1 Nash Equilibrium

A strategy profile $r^* = (r_1^*, r_2^*, ..., r_N^*)$ is a Nash equilibrium if and only if

$$
C_i(r_i^*, r_{-i}^*) \leq C_i(r_i, r_{-i}^*) \quad \forall r_i \in \{1, 2, ..., K\} \forall i. \tag{3.4}
$$

Note that apart from the load of processing queries generated by the peers that belong to the super-peer, and maintaining their caches, the super-peer also has to bear the cost of processing queries that arrive at it from other super-peers. The latter depends on the “centrality” of the super-peer in the overlay network graph. For our study, we assume this cost is uniform across all the super-peers and hence it is not considered in the local cost function.
Thus, at Nash equilibrium no peer will be able to improve its cost by unilaterally changing its current super-peer \( r^*_i \), \textit{i.e.}, provided that the decisions of all other peers are given by the assignment vector \( r^*_{-i} \).

\textbf{Theorem 1.} For the game described above, a Nash Equilibrium exists in pure strategies.

\textit{Sketch of Proof:} To construct a proof of Theorem 1, consider the following combinatorial optimization problem:

\[
\min_r C_0(r)
\]

where

\[
C_0(r) := \sum_i C_i(r_i, r_{-i}) = \sum_i \left( \frac{b_i}{w_{r_i}} \sum_{j : j \neq i, j \neq r_i} b_j + \frac{\mu}{2} \sum_{j : j \neq r_i} c_{ij} \right).
\] (3.5)

As the sum of the costs of all peers for a particular assignment vector, \( C_0 \) is also referred to as the \textit{social welfare} of the system in the game theoretic literature. Since this problem is combinatorial in nature, at least one globally optimum (minimum) solution exists for this problem. We denote such an assignment as \( \tilde{r} = (\tilde{r}_1, \tilde{r}_2, ..., \tilde{r}_N) \). In the Appendix, we will prove, by contradiction, that \( \tilde{r} \) is also a Nash equilibrium for our game, \textit{i.e.}, the Nash equilibrium is \( C_0 \)-“efficient”. Here, we sketch the main proof strategy. To wit, assume that \( \tilde{r} \) is not a Nash equilibrium. Then we can find at least one peer, say peer \( l \), which can improve its local cost given by (3.1), by changing its currently assigned super-peer \( k_1 \) to super-peer \( k_2 \), resulting in a new assignment vector \( r^* \). The change in global cost, in going from \( \tilde{r} \) to \( r^* \) by changing peer \( l \)’s assignment is \( C_0(r^*) - C_0(\tilde{r}) = 2\left( C_l(r^*) - C_l(\tilde{r}) \right) \) or \( C_0(r^*) = C_0(\tilde{r}) + 2\left( C_l(r^*) - C_l(\tilde{r}) \right) < C_0(\tilde{r}) \) (as shown in the Appendix), thus contradicting our assumption that \( \tilde{r} \) is the global minimum solution.

\section*{3.5 An Alternative Cost Framework}

Next, we state a centralized problem that reasonably models the system objective of load balancing and semantic proximity-based clustering. Let \( X \) be a \( K \times |V| \)
matrix such that \( x_{ki} = 1 \) if peer \( i \) belongs to a super-peer \( k \); otherwise \( x_{ki} = 0 \).

We require \( \sum_k x_{ki} = 1 \) \( \forall i = 1, 2, \ldots, |V| \), \( i.e., \)

\[
x_{ki} = 1 \quad \text{when } r_i = k.
\]  

(3.6)

Let \( w_k \) be the normalized capacity of the \( k \)th super-peer so that \( \sum_k w_k = 1 \). Then, the centralized super-peer assignment problem is:

\[
\min_{\tilde{X}} \tilde{C}_0(\tilde{X}) = \sum_{k=0}^{K} \left( \frac{\sum_{j \in V} x_{kj} b_j}{w_k} - \sum_j b_j \right)^2 + \frac{\mu}{2} \sum_{i,j} c_{ij} x_{ki} (1 - x_{kj})
\]  

(3.7)

subject to \( \sum_k x_{kj} = 1 \) \( \forall j \) and \( x_{kj} \in \{0, 1\} \) \( \forall k, j \).

The above standard formulation of the graph partitioning problem, e.g. [44] is a quadratic integer programming problem, the convexity of which depends on the network graph. In most cases it will not be convex. It would be ideal to decompose this problem into a set of \( K \) subproblems each of which is solved by a single partition. However, with the constraints \( \sum_k x_{kj} = 1 \) \( \forall j \) such a decomposition is difficult to realize. So, instead we study the effect of sequential peer-by-peer transfer on (3.7).

Let us consider an alternative local cost function for the graph partitioning problem.

\[
\tilde{C}_i(r_i, r_{-i}) = \frac{b_i^2}{w_{r_i}^2} + \frac{2b_i}{w_{r_i}} \sum_{j : j \neq r_i} b_j - \frac{2b_i}{w_{r_i}} \sum_j b_j + \frac{\mu}{2} \sum_{j : j \neq r_i} c_{ij}.
\]  

(3.8)

**Theorem 2.** For the local peer cost function (3.8), Nash equilibria exist at the local optima (maxima) of the centralized cost function (3.7).

**Proof.** : The proof is straightforward and directly follows from the definitions of the centralized cost and peer cost functions. Suppose that peer \( i \) is moved from super-peer \( m \) to super-peer \( n \), resulting in the new assignment matrix \( \tilde{X} \). Thus,
$\tilde{C}_0(\hat{X})$ is the new value of the cost function. We can directly show:

$$\tilde{C}_0(X) - \tilde{C}_0(\hat{X}) = \left(\frac{b_i^2}{w_m} + \frac{2b_i}{w_m} \sum_{j: r_j = m, j \neq i} b_j - \frac{2b_i}{w_m} \sum_{j} b_j\right) + \frac{\mu}{2} \sum_{j: r_j \neq m} c_{ij}$$

$$- \left(\frac{b_i^2}{w_n} + \frac{2b_i}{w_n} \sum_{j: r_j = n, j \neq i} b_j - \frac{2b_i}{w_n} \sum_{j} b_j\right) - \frac{\mu}{2} \sum_{j: r_j \neq n} c_{ij}$$

$$= \tilde{C}_i(n, r_{-i}) - \tilde{C}_i(m, r_{-i}), \quad (3.9)$$

or

$$\tilde{C}_0(X) = \tilde{C}_0(\hat{X}) + \tilde{C}_i(n, r_{-i}) - \tilde{C}_i(m, r_{-i}), \quad (3.10)$$

From (3.11), one can reduce the global cost $\tilde{C}_0$ iff there is a peer node $i$ that can modify its assignment to reduce its local cost defined by $\tilde{C}_i$. Thus one is at a local minimum of $\tilde{C}_0$ iff the assignment vector is at a Nash equilibrium with respect to the local peer costs.

### 3.5.1 Comparison of the Two Cost Frameworks

The first obvious difference between the two cost frameworks is the additional term $\frac{2b_i}{w_{r_i}} \sum_{j} b_j$ present in the alternative cost framework. This term is the sum of the query rates of all the peers in the system. The first cost framework is limited to requiring query rate information about its current super-peer and about the super-peer that it is evaluating as a prospective new super-peer. This property can be useful in a scenario where peers do not have access to the information about all super-peers. In our implementation of the algorithm described in Section 6, we will see that a peer can “view” only a limited set of super-peers (as defined by its scope of view) that it evaluates as its prospective super-peer. Secondly, in the following we will empirically see that the two cost frameworks perform very closely in terms of minimizing their respective global costs. Moreover, for certain range of values of $\mu$, the local incentive based cost framework gives better values for both $C_0$ and $\tilde{C}_0$. We performed a simple numerical study on random graphs that represented a graph of peers with its node and edge weight representation as described previously.
Table 3.1: Comparison of two cost frameworks ($C_0$ and $\tilde{C}_0$)

<table>
<thead>
<tr>
<th>Trial Number</th>
<th>$C_0$</th>
<th>$\tilde{C}_0$</th>
<th>No. of iterations to converge</th>
<th>$C_0$</th>
<th>$\tilde{C}_0$</th>
<th>No. of iterations to converge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>457134</td>
<td>4363</td>
<td>42</td>
<td>463130</td>
<td>6692</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>461704</td>
<td>4826</td>
<td>99</td>
<td>471539</td>
<td>9405</td>
<td>79</td>
</tr>
<tr>
<td>3</td>
<td>456260</td>
<td>2920</td>
<td>29</td>
<td>461614</td>
<td>5300</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>472157</td>
<td>2451</td>
<td>15</td>
<td>475814</td>
<td>3976</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>456336</td>
<td>3322</td>
<td>69</td>
<td>463559</td>
<td>6390</td>
<td>51</td>
</tr>
</tbody>
</table>

We used Netlogo [9], a multi-agent simulator, to generate random graphs of 230 peers (LPs) to be assigned to 5 super-peers indexed from $\{1,..,5\}$. The degree of each peer was randomly varied from 3 to 6. We randomly generated peer and edge weights each with mean 5. Starting with a random initial assignment, we performed greedy improvements wherein we chose a super-peer and the “most dissatisfied” peer belonging to that super-peer (in terms of the cost criteria under consideration) was allowed to switch its super-peer. We chose the super-peers sequentially by their index. We performed two runs, one for each cost criterion, until convergence. For a fair comparison between the two frameworks, the initial peer-to-super-peer assignments and the sequence in which super-peers were selected was the same for both the runs, the only change being the cost criterion employed. Convergence was implied when there was no more improvement in the respective global cost. Table 4.1 shows the results for 5 different random realizations of the graph. The values of $C_0$ and $\tilde{C}_0$ given in the table are at convergence.

We observed that using one of the cost frameworks did not guarantee a descent in the global cost for the other framework and vice-versa, thus corroborating that the two cost frameworks are inherently different. However, we can see from Table 1 that for the 5 different realizations of network graph, the first framework performed better in terms of both global costs $C_0$ and $\tilde{C}_0$. We subsequently performed a batch simulation with 50 random realizations of the graph. For each realization we kept...
the normalized super-peer speeds $w_k$ and the relative weight $\mu$ constant. It was observed that in 49 runs out of the 50 runs, the $C_i$ framework performed better in terms of both global costs, whereas in 1 run out of 50 runs, the $\tilde{C}_i$ framework performed better, but even then only with respect to its own global cost function $\tilde{C}_0$. This phenomenon was observed for all values of $\mu$ from 1 to 50 in steps of 1. Suppose we define a $C_0$-discrepancy ($\tilde{C}_0$-discrepancy) as an iterative step that increases $C_0$ ($\tilde{C}_0$) while using $\tilde{C}_i$ ($C_i$) as the local criterion for peer transfer. Over 50 runs, we observed that the average number of $C_0$-discrepancies for each run was about 0.2 whereas the average number of $\tilde{C}_0$-discrepancies for each run was about 5.2. It is apparent from these observations that, typically, for the range of $\mu$ from 1 to 50, $C_i$ allows for more “breadth” of search than $\tilde{C}_i$ and thus converges to better solutions for both global costs. Note that $\tilde{C}_i$ needs to explicitly consider load balancing among super-peers due to its direct relationship to $\tilde{C}_0$ which is a multi-criteria objective function with an explicit load balancing term. By contrast, moves using $C_i$ are “less constrained” but achieve load balance eventually due to the implicit and latent incentive in the definition of $C_i$ given by (3.1). Also, as $\mu$ was increased, the number of runs where the $\tilde{C}_i$ framework performed better increased, but again only in terms of its own global cost ($\tilde{C}_0$). This is due to the fact that the two cost frameworks differ only in terms of the computational load balancing criterion and an increase in $\mu$ gives less relative weight to computational load.

Now, let us consider another perspective. Each objective function ($C_0$ and $\tilde{C}_0$) is a multi-criteria objective function that includes both semantic similarity and load balancing terms. Further, the two cost frameworks differ only in the form of the load balancing term, i.e. $C_0 = B + \mu S$ and $\tilde{C}_0 = \tilde{B} + \mu S$, where $B$ ($\tilde{B}$) and $S$ are the load balancing costs and semantic similarity costs respectively at equilibrium for the cost framework $C_0$ ($\tilde{C}_0$). By varying the parameter $\mu$ that gives the relative weight to the two cost terms, one can explore the cost frontiers over both $(S,B)$ and $(S,\tilde{B})$ that are achieved by optimizing over $C_0$ ($\tilde{C}_0$). Fig. 3.1 and Fig. 3.2 show both the $(S,B)$ and $(S,\tilde{B})$ operating point tradeoffs achieved when optimizing over $C_0$ ($\tilde{C}_0$). Note that, for both plots, the value of $S$ depends on both the given value of $\mu$ and on the local minimum to which a given algorithm converges. So, for these plots we chose the range of $S$ (for the first plot) and $\tilde{S}$
(for the second plot) that approximately overlap with each other while using the two cost frameworks. We can see from Fig. 3.1 that the incentive based cost mechanism performs better in optimizing $B$ over the range of resulting semantic similarity cost, whereas, as seen from Fig. 3.2, we see that the alternative cost mechanism may not always be better in terms of optimizing $\tilde{B}$, i.e., for some values
of $\mu$ the incentive cost based mechanism can perform better in terms of both $B$ and $\tilde{B}$. For most of the $S$ range from 1100 to 1400, minimizing $C$ gives better results as seen in Fig. 3.2.

### 3.6 Local Incentive-based Algorithm

We will now describe the algorithm in detail. When a peer decides to relocate to a new super-peer, it evaluates the value of $C_i(r_i, r_{-i})$ for all $r_i \in R_i$ where $R_i \subset \{1, 2, ..., K\}$ is the set of super-peers within the peer’s scope of view. In this case, the scope of view of the peer depends on the history of transactions, i.e., the peers it has interacted with in the past. Although the formulation of the game in the previous section assumes that each peer makes the decisions based on the complete knowledge of the decisions of other peers, we will see in this section that we can relax this assumption and still converge to a fixed point in the peer strategy space. Particularly, we can show that when only a single peer migration (join or leave) is allowed per super-peer at a single time instance, then a decrease in the global cost function is guaranteed. This can be made possible with the use of a remote mutual exclusion (mutex) lock at the two super-peers involved in the transfer of the peer.

When a query is resolved, the super-peer that helped to resolve it embeds additional information in the response. This information helps the querying peer update its scope of view and consequently make an informed decision about migrating to a new super-peer. Particularly, the response to a resolved query contains the current load at the super-peer (say $k$) and a list of all peers that are currently assigned to it. Upon receipt of the resolved query, the querying peer updates its tables in the following way:

1. It adds the peer that resolved the query to the *proximity table* with its *query-hit-cnt* = 1, if not already present in the table. If it is already present in the table, it increments its *query-hit-cnt* by one.

2. It updates the *load* in *proximity table* for the super-peer that resolved the query.
3. It verifies that each peer in proximity table, that has its corresponding super-peer entry equal to the super-peer that helped resolve the query, still belongs to the super-peer.

The last step is done to ensure that value of peer assignments are up-to-date. Each peer can independently decide to update its super-peer and look for potential targets within its scope of view, i.e, those listed in the super-peer table. In order to ensure decision is made based on up-to-date information, a super-peer is considered as a potential target only when it resolves a query. The algorithm below describes the steps for peer migration.

1: repeat
2: Wait
3: until Query is Resolved
4: Update the proximity table using the data from query resolved in super-peer k.
5: Calculate and update the values in super-peer table for the super-peer k.
6: Compute $C_i(k, r_i)$ and $C_i(r_i, r_{-i})$.
7: if $C_i(k, r_i) < C_i(r_i, r_{-i})$ then
8: Acquire mutex lock at super-peer $r_i$.
9: Acquire mutex lock at super-peer k.
10: Change super-peer, i.e., $r_i = k$.
11: Release both mutex locks.
12: end if

Figure 3.3: Steps for Peer Migration

The transfer of a peer to a new super-peer happens only after a query is resolved at the new super-peer. This ensures that the peer has the latest knowledge about the super-peer that it is evaluating as its prospective super-peer. We assume that each super-peer has a mutex lock that allows only one peer to be added or removed from its cluster at a time. Such a mutex lock can be implemented using a remote inter-process synchronization scheme. In order for the peer to switch its super-peer, it needs to acquire mutex locks at both the current and its future super-peer. In order to prevent frequent migration of peers from one super-peer to another, the peer can evaluate a potential super-peer for migration asynchronously only at specific time instances. The above algorithm allows for such asynchronous transfers and still guarantees descent in the global cost.
3.6.1 Convergence

We can prove that under the assumption that only a single peer joins or leaves a super-peer cluster at a time, the algorithm converges to a locally optimal point. This is because the transfer of a peer affects the cost of only those super-peers that reside in one of the two super-peers involved in the transfer. Thus we can allow simultaneous transfer of peers in an asynchronous manner under the above assumption.

Theorem 3. The clustering algorithm of Fig. 1 converges.

Proof. : The function \( C_0(r) \) defined in Theorem 1 will be shown to be a potential function. Suppose the \( l \)th peer is transferred from its current allocated super-peer \( r_l \) to a new super-peer \( r^*_l \). Call the new assignment vector \( r^* \). Then obviously,

\[
C_l(r^*) - C_l(r) < 0.
\]

From the proof of Theorem 1 in the Appendix (replacing the global optimum assignment vector \( \hat{r} \) with any arbitrarily chosen assignment vector \( r \)), we have:

\[
C_0(r^*) - C_0(r) = \left( C_l(r^*) - C_l(r) \right) + \left( \sum_{i:j \neq l, r_i^* = r_l^*} C_i(r^*) - \sum_{i:j \neq l, r_i = r_l} C_i(r) \right) + \left( \sum_{i:j \neq l, r_i^* \neq r_l^*} C_i(r^*) - \sum_{i:j \neq l, r_i \neq r_l} C_i(r) \right)
= 2(C_l(r^*) - C_l(r)) < 0
\]

Hence for every local peer-level move, the potential function \( C_0(r) \) decreases and we know that due to bounded nature of the combinatorial optimization problem, there exists an assignment vector \( r \) that will yield a local minimum value of \( C_0 \) which is greater than or equal to the global minimum value, which is itself strictly bounded. Hence an achievable lower bound for the potential function exists and so, we can conclude that the algorithm converges.

Note that when a new peer joins the network, there is no initial information concerning its “proximity” with other peers in the network. Initially, all its edges
in the graph have zero weights. So it decides its super-peer based solely on its own query generation rate and the existing load on the super-peers. Also, since it cannot know the loads at each and every super-peer, it randomly samples a few super-peers and checks their existing load to decide which to join. It learns about its relationship with other peers over time in the same way that other peers learn about their relationship with the new peer. The $\mu$ parameter in (3.1) allows the peer to control the weight it gives to the cost of semantic similarity. When a node joins the system, it could start off with small $\mu$ and then gradually increase $\mu$ as it learns about its semantic relationships.

When a new super-peer is added to the system, its presence will remain unknown to all the peers until a newly arriving peer is attached to it. This is because a peer will know about the new super-peer only when its query is resolved by a peer attached to the new super-peer. This might delay convergence to the equilibrium system state. In order to speed up the process the newly arriving super-peer can advertise itself to its neighboring super-peers which in turn will forward the advertisement to their neighbors and so on. The peers upon learning about the new super-peer will add them to the super-peer table and evaluate for super-peer transfer. In the case of a super-peer failure, the local cost of all the “orphaned” peers becomes infinite. These peers then evaluate the super-peers in their super-peer table and select the one with minimum local cost. However, the information about some super-peers may not be up-to-date and hence the initial few transitions could be inconsistent, i.e., they may increase the global cost (3.5).

### 3.7 P2P File Sharing Experiment

In this section we describe our model for simulating the P2P file sharing network. We compared the performance of the schemes based on the two different cost criterion that we described above in terms of different performance parameters such as the average hop count and the time in clock ticks for query resolution, cache hit ratio, and a measure of semantic clustering. We also considered two additional simple schemes that use either the load balancing criterion (where the peer simply chooses the least loaded super-peer in its scope of view) or semantic proximity criterion (where the peer simply chooses the super-peer that is closest
semantically) for choosing the super-peer. These are used for comparison with the proposed incentive-based scheme that uses both. Moreover, it provides convergence guarantee. In our experiments, we evaluate the convergence by observing that the number of peer transitions per clock-tick gradually decreases to zero as the simulation progresses.

We modeled a discrete-time based simulator and at each tick of the global clock, events due to the generation or forwarding of queries were processed. The network and traffic model that we employed in our simulations was based on the probabilistic models described in [45]. These models have a direct effect on the query generation and resolution mechanisms in the file sharing system. The “Query-Cycle Simulator” described in [45] is based on observations from real P2P file sharing networks such as Gnutella. It provides probabilistic models of the semantic-type distribution and the file/content distribution within each semantic type based on semantic and file popularity observed in real P2P networks and the Web. The super-peer overlay network was unstructured and the Gnutella-like model of network overlay that we employed used a hop-count-restricted randomized query search. Each peer generated a query at its super-peer and the super-peer was responsible for forwarding/resolving the query. Each super-peer maintained a list of queries that needed to be serviced (pending-qry-list), a list of peers that were currently assigned to it (peer-list) and a list of neighboring super-peers (neighbor-list). The processing speed of each processor was given by proc-speed which indicated the number of queries that it could process per clock-tick. The relative measure of proc-speed determined the parameter $w_k$ in (3.1) for each super-peer $k$. Each peer node was assigned random rate of query generation qry-rate uniformly chosen with a given mean and variance. In our experiments, a peer was assigned a rate between 5 to 8 queries per 1000 clock ticks. This rate was chosen in order to minimize the chance of overflowing the query buffer at any super-peer and at the same time ensuring that super-peers are not lightly loaded.

The files in the system were categorized into 500 semantic categories, with each category $s_i$ ordered according to its popularity, i.e., category $s_i$ is the $i^{th}$ most popular category. Each semantic category had a set of 1000 files which were also indexed according to their popularity, i.e., file $f^j_i$ is the $j^{th}$ most popular file within the category $s_i$. The popularity of the semantic category was modeled as a Zipf
distribution [46], i.e., the probability that a file in semantic category $s_i$ is queried by a peer or available at a peer was proportional to $1/i$. Similarly, the popularity of files within each category was also modeled by the Zipf distribution. Each peer $k$ chose a set $C_k$ of 4 distinct file categories according to the Zipf distribution as described above. The peer generated queries and was able to resolve queries only for files belonging to these 4 semantic categories. For each semantic category $s_i$, a peer assigned a weight $a_k^i$ such that $a_k^i = 0$ if $s_i \notin C_k$ and $\sum_i a_k^i = 1$. So, $a_k^i$ gives the probability that a peer queries for a file belonging to category $s_i$. The probability that a peer $k$ queries a file $f_j$ belonging to category $s_i$ is given by

$$p_k^{ij} = a_k^i \frac{1/j}{\sum_{h=1}^{1000} 1/h}. \quad (3.12)$$

The above equation is also used to find the probability that the queried file is resolved by peer $k$.

We simulated a file-sharing P2P network consisting of 100 super-peers and 10000 peers that connect themselves to one of these super-peers. The initial super-peer for a peer was chosen uniformly at random. For each super-peer, proc-speed was initialized based on uniform random distribution and then normalized. The files were classified into 500 content categories (or semantic types) as described above. These figures were arbitrarily chosen; however they closely resemble realistic networks parameters. The overlay network of super-peers was formed, where each super-peer randomly chose its neighbors. The number of neighbors for each super-peer was a random number varying uniformly from 3 to 8. The super-peer architecture was fixed and we considered random query forwarding similar to Gnutella-like networks. Specifically, a super-peer, after finding that it is unable to satisfy the query, forwards it to a randomly chosen neighbor (other than the one from whom it received the query). The query survives in the network for a limited number of hops (TTL) after which it is terminated. At each clock-tick, a peer decides to generate a query based on its qry-rate. The file that is queried is decided by the probability in (3.12). The query is added to the pending-qry-list of the super-peer. At each clock-tick, a super-peer processes a number of queries in its pending-qry-list depending on its proc-speed. For each query being processed, the super-peer checks if the query could be resolved by one of its peers using the dis-
tribution given in (3.12). If the query is resolved then it is reverse-path forwarded back to the querying peer. Note that such reverse-path forwarding schemes, as in the case of onion routing [47], are commonly used to protect the privacy of the querying and the queried peer in P2P systems. Every query resolution at a peer initiates a process of updating the proximity tables, wherein the scope of view of the peer is updated. Based on the updated tables and the local frequency of update (i.e., transfer-rate, given by the number of resolved queries at the peer between transfers), the peer decides to transfer to a new super-peer using one of the schemes. Our goal was to determine the performance of these different schemes for various figures of merit.

3.8 Numerical Results

Our first set of experiments dealt with the choice of $\mu$, the relative weight given to the semantic proximity cost. We kept the transfer-rate at 10, i.e., a peer decided to seek a new super-peer once every 10 resolved queries. In Fig. 3.4 and Fig. 3.5, we can observe that with an initial increase in $\mu$, average query hop count and the average query resolution time decreased. The decrease here is sharp because as the semantically similar peers are clustered together, the inter-super-peer traffic is reduced, reducing the computation load on the super-peers and thus giving a better average query resolution time. For a critical value of $\mu$ (in this case, 0.15), we get a minimum value of average query resolution time and average hop count, after which the average query resolution time increases and the average hop count remains approximately constant. This is because as $\mu$ increases, more emphasis is given on clustering similar peers together, as compared to load balancing. Hence, in the case of peers belonging to the more popular semantic types, they try to cluster together, increasing the load on the super-peer and thus deteriorating the query resolution time. Similarly, we found the optimum value of $\mu$ for the alternative cost based decision in (3.8) and found out it to be around 160. Note that the average values were calculated over a set of 25 trials.

For our next experiment, we compared the four methods: incentive-based, alternative centralized cost based, purely load balancing based, and purely semantic clustering based. The comparisons were made based on the number of hop counts
and time (in clock-ticks) to resolve the query. The transfer-rate was varied. We ran the simulation for 20000 clock-ticks and averaged the values over 25 runs. We can observe that in terms of the average query resolution time, the incentive based scheme performs slightly better than the alternative cost based scheme and almost equally well in terms of the number of hops. This is due to the fact that the number of hop counts needed to resolve the query depends only on the semantic proximity cost and by observing (3.1) and (3.8), both methods have similar semantic proximity cost components. If the performance of an actual system (in terms of the
average query processing time) depends heavily on the load balancing component, then the performance gap between the two methods will increase. Also, recall that the alternative cost function needs an additional global parameter (sum of the rates of the peers) which needs to be feasibly communicated across all peers and will incur some cost when peer churn is high. As seen in Fig. 3.6 and Fig. 3.7, we can see that the scheme based on pure load balancing performs worst in terms of both figures of merit and its performance is more or less independent of the transfer-rate. This is obvious since semantic proximity decides not only the average hop count, but also the query resolution time. The scheme based on purely semantic clustering performs better than others in terms of the average hop count, but is sub-optimal in terms of the query resolution time.

![Figure 3.6: Effect of transfer-rate (Average hop count)](image)

A primary advantage of using the proposed incentive based cost criterion is its convergence guarantee for static network parameters (i.e., when parameters of the peers and super-peers such as rate or capacity do not change and there is no peer churn or super-peer failure). To see this, we observed the number of peer transitions as the simulation progressed. We compare it with another method whose incentive is simply the addition of load and semantic proximity. We call this method simple-partition. For these methods we observe the number of peer transitions across the simulation time. As seen in Fig. 3.8, the proposed method converges much
Figure 3.7: Effect of transfer-rate (Average resolution time)

faster than simple-partition. Moreover, the simple-partition method allows non-zero residual transitions in steady state, i.e., it does not converge to equilibrium. For this particular experiment we emphasized the cost of load balancing more than that of semantic proximity cost by keeping \( \mu \) very small because the two methods only differ in the way the load balancing term is incorporated in the cost function.

Figure 3.8: Convergence time for static network parameters

Next, we studied the effect of super-peer churn and failure and its effect on
system stability and recovery time. We ran the simulation for 2000 clock ticks until a steady state was reached, i.e., when no more peer transitions were observed. At this point in time, one of the super-peers withdrew which made the local cost of all the peers attached to this super-peer infinite as shown in Fig. 3.10. These “orphaned” peers then evaluated super-peers in their super-peer table to join. A new super-peer was simultaneously added with similar resource capacity as the failed super-peer. This super-peer advertised itself as described at the end of Section 6. We observed the number of peer transitions for every 50 clock-ticks. As seen in Fig. 3.9, the system took approximately 400 clock-ticks to recover and converge to equilibrium state. Fig. 3.10 shows the social welfare cost function (3.5) during super-peer churn. We observed the social welfare cost after every interval of 50 clock-ticks.

![Figure 3.9: Convergence time after super-peer churn](image-url)
Figure 3.10: Social welfare cost during and after super-peer churn
Chapter 4

Multicategory Crowdsourcing
Accounting for Plurality in Worker Skill and Intention and Task Difficulty

4.1 Crowdsourcing: An Introduction

Crowdsourcing systems leverage the diverse skill sets of a large number of Internet workers to solve problems and execute projects. In fact, the Linux project and Wikipedia can be considered products of crowdsourcing. These systems have recently gained much popularity with web services such as Amazon MTurk and Crowd Flower, which provide a systematic, convenient and templatized way for requestors to post problems to a large pool of online workers and get them solved quickly. The success of crowdsourcing has been demonstrated for annotating and labeling images and documents [48], writing and reviewing software code, designing products [49], and also raising funds. Here, we focus on crowdsourcing tasks with a categorical answer space.

www.mturk.com
www.crowdflower.com
www.topcoder.com
www.crowdfunding.com
Although the crowd expedites annotation, its anonymity allows noisy or even malicious labeling to occur. Online reputation systems can help reduce the effect of noisy labels, but are susceptible to Sybil [50] or whitewashing [51] attacks. Moreover, the aggregate reputation score only reflects a worker’s skill on previous tasks/domains. This may not be a good indication of his skill on new domains, for which he has not been evaluated. A second way to mitigate worker unreliability is to assign each task to multiple workers and aggregate their answers in some way to estimate the ground truth answer. The estimation may use simple voting or more sophisticated aggregation methods, e.g., [52], [53], [54]. The aggregation approaches we propose in this work have several notable characteristics. First, we make a clear distinction between worker skill level and worker intention, i.e., whether the worker is honest or malicious. This dissection allows us to plausibly characterize the behavior of an adversarial worker in a multicategory setting. Our aggregation approaches explicitly identify such workers and exploit their behavior to, in fact, improve the crowd’s accuracy (relative to the case of strictly non malicious workers). Second, most approaches are only suitable for binary (two choice) tasks. By contrast, our approaches explicitly handle multicategory tasks, including binary as a special case. Third, and most importantly, some crowdsourcing methods [55], [54] exploit and thus rely on certain biases in the ground-truth answering mechanism, or in the worker answering mechanism to achieve good performance. Supposing the categorical answer space is \{a, b, c, d, e\}, and that there is a batch of \(T\) tasks to be jointly solved by the crowd, several notable examples are:

1. A worker, when guessing, may \(e.g.,\) be more likely to choose the last answer \(e\), than the first \(a\).

2. The ground-truth answer may be more likely to be, \(e.g.,\) in position \(b\) than in position \(e\).

3. All the tasks in the batch may come from the same classification domain, with answer \(a\) statically (always) corresponding to a particular class on the domain, \(b\) to a different class and so on for \(c, d\) and \(e\). If the classes are not equally likely then this can be exploited.
Some methods are specifically designed to exploit these types of biases in the answer space. However, we note that such biases can be easily removed, e.g., by randomly reordering the set of answers for every task in the data batch. Moreover, considering example 3, it is possible that the batch does not consist of tasks from a single classification domain, but rather from a mix of multiple domains. For example, workers with good visual acuity who are also world travellers may be given a batch of images, and asked to identify country of origin for some images, crop type for others, or whether there are vehicles present in the third subset of tasks. Moreover, each task in the batch may not be annotated by the underlying “domain” (subset). In this example, there are three domains (detecting country, crop type and vehicle). In this heterogeneous setting, answer a does not correspond to a particular category on a single domain. Thus it is not possible in this setting, to exploit single-domain, non uniform class priors in the crowd aggregation approach. The methods we will develop do not assume and thus rely on the above sources of bias to achieve good performance. Instead, we will exploit diversity in worker skill and task difficulty, and our capability to accurately infer these parameters, given a batch of tasks. We also emphasize that it is possible to extend our approaches to also exploit the above sources of bias, if they may exist. Finally, many crowdsourcing approaches use hyper-parameters and require properly setting them, which is very difficult in an unsupervised setting (where there are no “probe” tasks in the batch with the ground-truth answers known). By contrast, our approaches do not require any hyper-parameters. We propose two distinct frameworks for jointly estimating workers’ skills, intentions, and the ground truth answers for the tasks.

Our first approach assumes a stochastic model for answer generation that plausibly captures the interplay between worker skills, intentions, and task difficulties. To the best of our knowledge, this is the first model that uses a common metric space (on the real line) to represent task difficulties and worker reliabilities and this allows us to directly compare worker skill and task difficulty, which we exploit in our stochastic model. In this model, we formalize the notion of an adversarial worker and discuss and model different types of adversaries. A simple adversary gives incorrect answers “to the best of his skill level”. More “crafty” adversaries can attempt to evade detection by only giving incorrect answers on the more difficult tasks solvable at their skill level. The detection of adversaries and the estimation of
both worker skills and task difficulties can be assisted by the knowledge of ground-truth answers for some (probe) tasks. Accordingly, we formulate a semisupervised approach, invoking a generalized EM (GEM) algorithm [56] to maximize the joint log likelihood over the (known) true labels for the “probe” tasks and the answers of the crowd for all tasks. This general approach specializes to an unsupervised method when no (labeled) probe tasks are available. Our numerical study considers both semisupervised and unsupervised flavors of our method on a variety of data sets. Interestingly, our crowd aggregation rule comes precisely from the E-step of our GEM algorithm, since the ground-truth answers are treated as the hidden data [15] in our GEM approach.

A limitation of the stochastic modeling approach is that, on some domains, its statistical assumptions may not well-characterize crowd behavior. Accordingly, in this work we also investigate “deterministic” approaches that do not assume underlying stochastic answer generation mechanisms and hence are expected to perform robustly across a diverse set of domains. We propose two deterministic objective-based methods that jointly estimate worker intention, skill and the ground truth answers to multicategory tasks by maximizing a measure of aggregate confidence on the estimated ground truth answers measured over the batch of tasks. Here, crowd aggregation is achieved by weighted plurality voting, with higher weights given to the workers estimated to be the most highly skilled and honest. These workers are identified essentially by their tendency to agree with each other on most of the tasks, unlike the low skilled workers, who tend to answer arbitrarily.

Probe tasks are a form of overhead which limits the number of true tasks in a batch (of fixed size) that the crowd is solving. It may be expensive, time-consuming and/or impractical to devise meaningful probe tasks for a given problem domain. Accordingly, we consider our methods in both semisupervised and unsupervised settings to evaluate the gains obtained by using probes. Our experimental evaluation of the proposed schemes consisted of three levels. First, we evaluated the schemes using simulated data generated in a way consistent with our proposed stochastic generative model. This allowed us to study the robustness of our GEM algorithm by comparing estimated and actual model parameters. Second, we evaluated and compared performance of the methods using a crowd of “simulated” workers that do not obviously generate answers in a fashion closely matched to
our model. Specifically, each worker was a strong learner, formed as an ensemble of weak learners. Each weak learner was a decision tree, with the ensemble (and thus, a strong learner) obtained by multiclass boosting. A strong worker’s skill was controlled by varying the number of boosting stages used. We performed experiments on UC Irvine data sets [57] and studied the comparative gains of our methods over benchmark methods on a variety of classification domains. Our final experiment involved a crowdsourcing task we posted using Amazon Mturk. Overall, we observed that our methods are especially advantageous when there is an (a priori unknown) minority of skilled workers amongst a large crowd of unskilled (as well as malicious) workers, i.e., they are able to overcome “tyranny of the masses” by estimating the workers’ intentions and skills with a reasonable level of accuracy.

4.2 Generative Semisupervised Modeling Approach

In the following framework, we separately model worker intention and skill. We represent a worker’s intention using a binary parameter indicating if he is adversarial or not. An honest worker provides accurate answers “to the best of his skill level” whereas an adversarial worker may provide incorrect answers “to the best of his skill level”. In the case of binary crowdsourcing tasks, adversarial workers can be identified by a negative weight [53] given to their answers. Here we extend malicious/adversarial worker models to multiclass tasks and hypothesize both “simple” and “crafty” adversaries.

Our approach incorporates task difficulty and worker skill explicitly and, unlike previous approaches [58] [48] [53], characterizes the interplay between them. Task difficulty and worker skill are both represented on the real line, with our generative model for a worker’s answer based on their difference. If the task difficulty exceeds a worker’s skill level, the worker answers randomly (whether honest or adversarial). For an adversary, if the task difficulty is less than his skill level, he chooses randomly only from the set of incorrect answers. We also acknowledge another category of worker type known as “spammers”. These are lazy workers who simply answer randomly for all tasks. Our model well-characterizes such workers via
large negative skill values.

4.2.1 Notation

Suppose a crowd of \(N\) workers is presented with a set of \(T_u\) unlabeled tasks, for which the ground truth answers are unknown. There are also \(T_l\) probe tasks, with known ground truth answers. We assume the crowd is unaware which tasks are probes. Accordingly, a malicious worker cannot alter his answering strategy in a customized way for the probe tasks to “fool” the system. Let \(\{1, 2, ..., T_l\}\) be the index set of the probe tasks and \(\{T_l + 1, T_l + 2, ..., T_l + T_u\}\) be the index set for non-probe tasks. We assume without loss of generality that each worker is asked to solve all the tasks. The answers are chosen from a set \(C := \{1, 2, ..., K\}\). Let \(z_i \in C\) be the ground truth answer and let \(\tilde{d}_i \in (-\infty, \infty)\) represent the difficulty level of task \(i\). The (ground truth) intention of worker \(j\) is indicated by \(v_j \in \{0, 1\}\), where \(v_j = 1\) denotes an honest worker and \(v_j = 0\) an adversary. \(d_j \in (-\infty, \infty)\) represents the \(j^{th}\) worker’s (ground truth) skill level and \(a_j\) denotes an additional degree of freedom to introduce variation in the probability mass function across workers (discussed in Section 4.2.3). Finally the response provided to the \(i^{th}\) task by the \(j^{th}\) worker is denoted \(r_{ij} \in C\).

4.2.2 Stochastic Generation Model

We define our model’s parameter set as \(\Lambda = \{(v_j, d_j, a_j) \ \forall \ j, \{\tilde{d}_i \ \forall \ i\}\}\). We hypothesize the generation of the answers for non-probe tasks in two steps. Independently for each non-probe task \(i \in \{T_l + 1, ..., T_l + T_u\}\):

1. Randomly choose the ground truth answer \((z_i)\) from \(C\) according to a uniform pmf \(\frac{1}{K}\).

2. For each worker \(j \in \{1, ..., N\}\), generate \(r_{ij} \in C\) for task \(i\) based on the parameter-conditional probability mass function (pmf) \(\beta(r_{ij}|\Lambda_{ij}, z_i)\), where

---

The unsupervised setting is a special case where \(T_l = 0\). We only make this assumption for notational simplicity. Our methodology in fact applies generally to the setting where each worker solves only a subset of the tasks.

One can always randomize the indexing of the answers for every task to ensure that the true answer is uniformly distributed over \(\{1, 2, ..., K\}\). This would remove any source of bias (e.g., toward the true answer being the first (1) or the last (K)).
\[ \Lambda_{ij} := \{v_j, d_j, a_j, \tilde{d}_i\}. \]

Also, independently for each probe task \( i \in \{1, \ldots, T\} \) and each worker \( j \), generate the answer \( r_{ij} \in C \) based on the parameter-conditional pmf \( \beta(r_{ij}|\Lambda_{ij}, z_i) \).

### 4.2.3 Worker Types

We model the ability of a worker to solve the task correctly using a sigmoid function based on the difference between the task difficulty and the worker’s skill, i.e., the probability that worker \( j \) can solve task \( i \) correctly is \( \frac{1}{1+e^{-a_j(d_j - \tilde{d}_i)}} \). Note we have included a degree of freedom \( a_j \) which attempts to capture the individuality of workers. It is also possible to tie this parameter, i.e., set \( a_j = a, \forall j \).

#### 4.2.3.1 Honest Workers

For an honest worker \((v_j = 1)\), the pmf \( \beta \) is defined as:

\[
\begin{align*}
\beta(r_{ij} = l|\Lambda_{ij}, v_j = 1, z_i) = \\
\begin{cases}
\frac{1}{1+e^{-a_j(d_j - \tilde{d}_i)}} + \left( \frac{1}{K} \right) \left( \frac{e^{-a_j(d_j - \tilde{d}_i)}}{1+e^{-a_j(d_j - \tilde{d}_i)}} \right) & \text{for } l = z_i \\
\left( \frac{1}{K} \right) \left( \frac{e^{-a_j(d_j - \tilde{d}_i)}}{1+e^{-a_j(d_j - \tilde{d}_i)}} \right) & \text{otherwise}
\end{cases}
\end{align*}
\]

Here, the worker essentially answers correctly with high probability if \( d_j > \tilde{d}_i \), and with probability \( \frac{1}{K} \) otherwise. Note that “spammer” workers, those with \( d_j < \min_i \tilde{d}_i \), will tend to answer randomly for all tasks, under this model. Next, we discuss two models for adversarial workers.

#### 4.2.3.2 Simple Adversarial Workers

For the simple adversarial model, \( \beta \) is given by

\[
\beta(r_{ij} = l|\Lambda_{ij}, v_j = 0, z_i) =
\]

The specific parametric dependence of \( \beta \) on \( \Lambda_{ij} \) will be introduced shortly. Alternative (soft) generalized step functions could also in principle be used here.
Here, essentially, the worker only chooses the correct answer (randomly) if the task difficulty defeats his skill level; otherwise he excludes the correct answer and chooses randomly from amongst the remaining answers. We describe such an adversary to be “simple” since he does not consider the possibility of being identified by a clever aggregation rule (such as the one proposed in this thesis). A more “crafty” adversarial worker will try to be more evasive. We describe one such model of an adversary in Section 4.6.

4.2.4 Incomplete, Complete and Expected Complete Data Log Likelihood

The observed data \( \mathcal{X} = \mathcal{R} \cup \mathcal{Z}_L \) consists of the set \( \mathcal{R} \) of answers given by the workers to all the tasks, i.e., \( r_{ij} \) \( \forall \ i, j \) and \( \mathcal{Z}_L = \{ z_i | i \in \{ 1, 2, ..., T_l \} \} \) is the set of ground truth answers to the probe tasks. We express \( \mathcal{R} = \mathcal{R}_L \cup \mathcal{R}_U \), i.e., the union of answers to probe tasks and non-probe tasks. We choose the hidden data \([15]\) to be the ground truth answers to the non-probe tasks, i.e., \( Z_i, i \in \{ T_l+1, ..., T_l+T_u \} \).

Based on the stochastic model in Section 4.2.2, the incomplete data log-likelihood, which we seek to maximize in estimating \( \Lambda \), is given by

\[
\log \mathcal{L}_{inc} = \log P(\mathcal{R}, \mathcal{Z}_L|\Lambda) + \log P(\mathcal{R}_L, \mathcal{R}_U|\Lambda) = \sum_{i=1}^{T_l} \sum_{j=1}^{N} \log \frac{1}{K} \beta(r_{ij}|\Lambda_{ij}, z_i) + \sum_{i=T_l+1}^{T_l+T_u} \sum_{j=1}^{N} \log \frac{1}{K} \sum_{k=1}^{K} \beta(r_{ij}|\Lambda_{ij}, Z_i = k)
\]

\[
\propto \sum_{i=1}^{T_l} \sum_{j=1}^{N} \log \beta(r_{ij}|\Lambda_{ij}, z_i) + \sum_{i=T_l+1}^{T_l+T_u} \sum_{j=1}^{N} \log \sum_{k=1}^{K} \beta(r_{ij}|\Lambda_{ij}, Z_i = k).
\]

Treating \( Z_i, i = 1, ..., T_l \) as the hidden data within the EM framework [15], the expected complete data log-likelihood, where the expectation is with respect to
the pmf \( P(Z_i = k|\mathcal{X}, \Lambda) \), can be written as:

\[
\mathbb{E}[\log L_c|\mathcal{X}, \Lambda] \propto \sum_{i=1}^{T_l+T_u} \sum_{k=1}^{N} \log \beta(r_{ij}|\Lambda_{ij}, z_i) + \sum_{i=T_l+1}^{T_l+T_u} \sum_{j=1}^{N} \sum_{k=1}^{K} [P(Z_i = k|\mathcal{X}, \Lambda) \log \beta(r_{ij}|\Lambda_{ij}, Z_i = k)]
\]

\[
= \sum_{i=1}^{T_l} \sum_{j: r_{ij} = z_i} \left[ v_j \log \left( \beta(r_{ij}|\Lambda_{ij}, v_j = 1, z_i = r_{ij}) \right) + (1 - v_j) \log \left( \beta(r_{ij}|\Lambda_{ij}, v_j = 0, z_i = r_{ij}) \right) \right]
\]

\[
+ \sum_{i=T_l+1}^{T_l+T_u} \sum_{k=1}^{K} P(Z_i = k) \left[ v_j \log \left( \beta(r_{ij}|\Lambda_{ij}, v_j = 1, Z_i = k) \right) \right] + \sum_{i=T_l+1}^{T_l+T_u} \sum_{k=1}^{K} \sum_{j: r_{ij} \neq k} P(Z_i = k) \left[ v_j \log \left( \beta(r_{ij}|\Lambda_{ij}, v_j = 1, Z_i \neq k) \right) + (1 - v_j) \log \left( \beta(r_{ij}|\Lambda_{ij}, v_j = 0, Z_i \neq k) \right) \right]
\]

4.2.5 The Generalized EM (GEM) Algorithm

We formulate our algorithm using the above defined expected complete data likelihood. The EM algorithm ascends monotonically in \( \log L_{inc} \) with each iteration of the E and M steps [15]. In the expectation step, one calculates the pmf \( P(Z_i = k|\mathcal{X}, \Lambda^t) \) using the current parameter values \( \Lambda^t \), and in the maximization step, one computes \( \Lambda^{t+1} = \arg \max_{\Lambda} \mathbb{E}[\log L_c|\mathcal{X}, \Lambda^t] \).

E step: In the E-step we compute the expected value of \( Z_u \) given the observed data \( \mathcal{X} \) and the current parameter estimates \( \Lambda^t \). Based on our assumed stochastic model (section 3.2), with data for each task generated i.i.d, we have that \( P(Z_u|\mathcal{X}, \Lambda^t) = \prod_{i=T_l+1}^{T_l+T_u} P(Z_i = z_i|\mathcal{X}, \Lambda^t) \). Moreover, again based on the assumed stochastic model
and applying Bayes' rule, we can derive the closed form expression for the pmf in
the E-step as:

\[ P_i(Z_i = k | \mathcal{X}, \Lambda^t) = \frac{\prod_{j=1}^{N} \beta(r_{ij} | \Lambda^t_{ij}, Z_i = k)}{\sum_{l=1}^{K} \prod_{j=1}^{N} \beta(r_{ij} | \Lambda^t_{ij}, Z_i = l)}, \quad \forall i \in \{T_l + 1, ..., T_u + T_l\}. \quad (4.3) \]

We can express the E step as an optimization problem [59] to find the probability
distribution that maximizes the following

\[ P(\mathcal{Z}_u | \mathcal{X}, \Lambda^t) = \arg \max_{\tilde{P}} \mathbb{E}_{\tilde{P}[\mathcal{L}]} + H(\tilde{P}) \quad (4.4) \]

where \( H(\tilde{P}) \) is the negative of the entropy for the pmf \( \tilde{P} \) given by

\[ H(\tilde{P}) = -\sum_{k=1}^{K} \tilde{P}(z_i = k | \mathcal{X}, \Lambda^t) \log (\tilde{P}(z_i = k | \mathcal{X}, \Lambda^t)) \quad (4.5) \]

We can rewrite the optimization problem as:

\[ P(\mathcal{Z}_u | \mathcal{X}, \Lambda^t) = \arg \max_{\tilde{P}} \left[ \sum_{i=T_l+1}^{T_u} \sum_{j=1}^{N} \log \beta(r_{ij} | \Lambda^t_{ij}, z_i) + N(T_l + T_u) \log \frac{1}{K} \right. \]

\[ + \left. \sum_{i=T_l+1}^{T_u} \sum_{j=1}^{N} \sum_{k=1}^{K} [\tilde{P}(z_i = k | \mathcal{X}, \Lambda^t) \log \beta(r_{ij} | \Lambda^t_{ij}, z_i = k)] + H(\tilde{P}) \right] \quad (4.6) \]

Note that the first two terms in the sum are independent of \( \tilde{P} \) and hence can
be ignored. Also note that due to independence, we can express

\[ \tilde{P}(\mathcal{Z}_u | \mathcal{X}, \Lambda^t) = \prod_{i=T_l+1}^{T_l+T_u} \tilde{P}_i(z_i | \mathcal{X}, \Lambda^t) \quad (4.7) \]

Hence we divide the optimization problem into \( T_u \) problems that can be solved
independently

\[
\begin{align*}
\text{P}_i(z_i | \mathcal{X}, \Lambda^t) &= \arg \max_{\tilde{P}} \left[ \sum_{j=1}^{N} \sum_{k=1}^{K} \tilde{P}(z_i = k | \mathcal{X}, \Lambda^t) \log \beta(r_{ij} | \Lambda_{ij}^t, z_i = k) \right] + H(\tilde{P}) \\
&= \arg \max_{\tilde{P}} \left[ \sum_{k=1}^{K} \tilde{P}(z_i = k | \mathcal{X}, \Lambda^t) \log \left( \prod_{j=1}^{N} \beta(r_{ij} | \Lambda_{ij}^t, z_i = k) \right) \right] \\
&= \arg \min_{\tilde{P}} \left[ \sum_{k=1}^{K} \tilde{P}(z_i = k | \mathcal{X}, \Lambda^t) \log \frac{\tilde{P}(z_i = k | \mathcal{X}, \Lambda^t)}{\left( \prod_{j=1}^{N} \beta(r_{ij} | \Lambda_{ij}^t, z_i = k) \right)} \right]
\end{align*}
\]

A unique solution exists for the above optimization problem given by

\[
\begin{align*}
\text{P}_i(z_i = k | \mathcal{X}, \Lambda^t) &= \frac{\prod_{j=1}^{N} \beta(r_{ij} | \Lambda_{ij}^t, z_i = k)}{\sum_{l=1}^{K} \prod_{j=1}^{N} \beta(r_{ij} | \Lambda_{ij}^t, z_i = l)} \\
\end{align*}
(4.9)
\]

**Generalized M step:** In the M-step of EM, one maximizes the expected complete data log-likelihood with respect to the model parameters:

\[
\Lambda^{t+1} = \arg \max_{\Lambda} \mathbb{E}[\log \mathcal{L}_c(\Lambda) | \mathcal{X}, \Lambda]. \quad (4.10)
\]

Since \( \Lambda \) consists of mixed (both continuous and discrete) parameters, with a particular parametric dependence and with \( 2^N \) (honest, adversarial) crowd configurations, it is not practically feasible to find a closed form solution to (4.10) for our model. Instead, we use a generalized M-step approach \cite{56}\cite{60} to iteratively maximize over the two parameter subsets \( \{v_j \forall j\} \) and \( \{(d_j, a_j) \forall j\}, \{\tilde{d}_i \forall i\} \).

**M1 Substep:** Since (4.3) is an additive function of terms that each depend on a single variable \( v_j \), we can find a closed form solution for \( v_j \forall j \) given all other parameters fixed:

\[
\tilde{v}_j = \arg \max_{v_j \in (0,1)} \mathbb{E}(\log \mathcal{L}_c(\{v_j\}) | \mathcal{X}_j, \tilde{\Lambda} \setminus \{v_j\}). \quad (4.11)
\]

Here \( \mathcal{X}_j \) is the set of answers provided by the \( j^{th} \) worker and the ground truth answers for the probe tasks that he answered and \( \tilde{\Lambda} \) is the result of the previous
M2 substep.

**M2 Substep:** We maximize $E[\log L_c(\Lambda \backslash \{v_j\})|\mathcal{X}, \{\tilde{v}_j\}]$ with respect to $\Lambda \backslash \{v_j\}$ given $\{\tilde{v}_j\}$ fixed from the previous M1 substep. For this, we use a gradient ascent algorithm which ensures monotonic increase in $\log L_{mc}$, but which may only find a local maximum, rather than a global maximum of $E[\log L_c(\Lambda \backslash \{v_j\})|\mathcal{X}, \{\tilde{v}_j\}]$. At convergence, the result is stored in $\tilde{\Lambda} \backslash \{v_j\}$. The M1 and M2 substeps are applied alternately, iteratively, until convergence. $\Lambda^{t+1}$ stores the result of the generalized M-step at convergence.

**Inference:** Note that the E-step (4.3) computes the *a posteriori* probabilities of ground-truth answers. Thus, after our GEM learning has converged, a maximum *a posteriori* decision rule applied to (4.3) gives our crowd-aggregated estimates of the true answers for the non-probe tasks.

### 4.2.6 Unsupervised GEM

Note that when probe tasks are not included in the batch of tasks, an unsupervised specialization of the above GEM algorithm is obtained. In particular, we have $T_i = 0$, with the first term in (4.3) and the first two terms in (4.3) not present. Our above GEM algorithm is accordingly specialized for this case. In Section 4.4, we will evaluate the unsupervised GEM based scheme along with all other methods.

### 4.3 Energy-Constrained Weighted Plurality Aggregation

Performance of our GEM approach will in general depend on how well the true answer generation mechanism resembles the stochastic model assumed in Section 4.2.2. We would also therefore like to explore an alternative “principle” on which to base crowd aggregation, without any explicit assumption about the underlying stochastic answer generation model. The methods we propose in this section use weighted plurality voting, where the weights assigned to workers essentially reflect their individual skill level. A key idea here is to make the weight vector “energy-constrained”, so as to obtain a bounded solution to the resulting optimiza-
tion problem. The methods we will propose are applicable to both unsupervised and semisupervised settings, but for clarity of presentation, we will focus on the unsupervised setting and then delineate how to extend these approaches to exploit probe tasks, if available.

4.3.1 From simple plurality to weighted plurality voting

We will introduce and define new variables to clearly explain our approach. Let $T = T_l + T_u$ and $\hat{z}_i$ be the $K \times 1$ vector representing the inferred ground truth answers with $\hat{z}_{im} \in \{0, 1\}$ and $\sum_m \hat{z}_{im} = 1$, i.e., $\hat{z}_{im}$ is 1 when the inferred answer to the $i^{th}$ task is $m$. Also, $\hat{Z} = (\hat{z}_i, i = 1, ..., T)$. All other definitions from section 4.2.1 will be used. A natural extension of majority voting to the multicategory case is plurality voting, where the answer that gets the maximum number of votes is the inferred answer for that task. To help motivate what follows, we note that plurality voting is the solution of a maximization problem defined over a given batch of tasks. In particular it solves:

$$\max_{\hat{Z}} \sum_{i=1}^T \sum_{m=1}^K \hat{z}_{im} \sum_{j=1}^N \delta(r_{ij} - m)$$

subject to

$$\hat{z}_{im} \in \{0, 1\}, \quad \sum_m \hat{z}_{im} = 1, \quad i = 1, ..., T,$$

where $\sum_j \delta(r_{ij} - m)$ is the total vote for answer $i$ by all workers.

Plurality-based voting is expected to perform well when all the workers are honest and “equally reliable”, with worker accuracy greater than that of a spammer. However, the crowd may consist of heterogeneous workers with varying skill levels and intentions, as we considered before in Section 4.2. For the most challenging but realistic “tyranny of the masses” case where a small proportion of highly skilled workers exist among a mass of unskilled workers or spammers, standard plurality-

Ties could be broken by randomly selecting from among the set of plurality answers.

The expected accuracy of the spammer is $\frac{1}{K}$, where $K$ is the number of possible answers for each task.
based voting will be highly sub-optimal. Even supposing the highly skilled workers are *always* correct, “one worker, one vote” means that, if the skilled worker subset is a small minority, it will not in general be able to “tip the balance” of the plurality towards the correct answer. Alternatively, here we consider weighted plurality voting schemes, where different weights are assigned to the workers based on their “accuracy level”, accounting for both intention and skill. Allocation of higher weights to the most skilled workers may allow defeating “tyranny of the masses”. Moreover, for weighted plurality voting, ties will almost never occur. In order to ensure well-posed optimization problems, we will impose an energy constraint on the weight vector. We will first propose to jointly estimate the worker weights and the ground truth answers consistent with solving the following optimization problem:

$$
\max_{\bar{Z}, w} \psi_{wp}(w, \bar{Z}) = \sum_{j=1}^{N} w_j \sum_{i=1}^{T} \sum_{m=1}^{K} \hat{z}_{im} \delta(r_{ij} - m)
$$

subject to

$$
\sum_{j=1}^{N} w_j^2 = 1, \quad \hat{z}_{im} \in \{0, 1\}, \quad \sum_{m=1}^{K} \hat{z}_{im} = 1.
$$

Here, we maximize the average weighted plurality score (a measure of aggregate confidence in decisionmaking). We will solve (4.13) in a locally optimal fashion via an iterative algorithm, alternating between the updates of the weights and the inferred answers. We iterate over the following two (local maximization) steps until convergence.

**Step 1**: For fixed $w$ and for each task $i$, the choice for $\hat{z}_i$ which maximizes (4.13) is :

$$
\hat{z}_{im} = \begin{cases} 
1 & \text{if } \sum_{j: r_{ij} = m} w_j > \sum_{j: r_{ij} = k} w_j \forall k \neq m \\
0 & \text{otherwise}
\end{cases}
$$

(4.14)
Step 2: Given fixed $\hat{Z}$ we compute the optimum $w$, maximizing (4.13). The Lagrangian for this optimization problem is

$$L = \sum_{j=1}^{N} w_j \sum_{i=1}^{T} \sum_{m=1}^{K} \hat{z}_{im}(\delta(r_{ij} - m) + \lambda(\sum_{j=1}^{N} w_j^2 - 1)). \quad (4.15)$$

Differentiating with respect to $w_k$, we get

$$\frac{\partial L}{\partial w_k} = \sum_{i=1}^{T} \sum_{m=1}^{K} \hat{z}_{im}\delta(r_{ik} - m) + 2\lambda w_k = 0 \quad (4.16)$$

$$\Rightarrow w_k = -\frac{1}{2\lambda} \sum_{i=1}^{T} \sum_{m=1}^{K} \hat{z}_{im}\delta(r_{ik} - m) \quad \forall k$$

We can compute $\lambda$ by squaring the above and then summing over all workers $j$, to find

$$\lambda = \frac{1}{2} \sqrt{\sum_{j=1}^{N} \left( \sum_{i=1}^{T} \sum_{m=1}^{K} v_{im}\delta(r_{ij} - m) \right)^2} \quad (4.17)$$

Hence the optimal value of $w_k$ is given by

$$w_k^* = \sqrt{\frac{\sum_{i=1}^{T} \sum_{m=1}^{K} \hat{z}_{im}(\delta(r_{ik} - m)} {\sum_{j=1}^{N} \left( \sum_{i=1}^{T} \sum_{m=1}^{K} \hat{z}_{im}\delta(r_{ij} - m) \right)^2} \quad \forall k. \quad (4.18)$$

Each of the above two steps ascends in the objective (reward) function $\psi_{wp}$, with convergence to a local maximum, starting from an “unbiased” initial weight vector $w = \epsilon 1$, where $\epsilon = \frac{1}{\sqrt{N}}$ and $1$ is an $N \times 1$ vector of all ones.

4.3.2 Accounting for Adversaries

Note that we have not accounted for adversarial workers in (4.13). To do so now, suppose that an adversarial worker $k$ will choose the incorrect answer randomly (uniformly over all incorrect answers). In the following we will develop two exten-
sions of the weight-constrained problem to accommodate the worker’s intention. Our first approach uses binary parameters to represent worker intentions, whereas our second method exploits the extra degree of freedom provided by the sign of the estimated weight \( w_j \) to represent the worker’s intention.

### 4.3.2.1 Introducing Binary Parameters to Represent Intention

Suppose we introduce an additional set of variables given by the \( N \times 1 \) vector \( v \), where \( v_j \in \{0, 1\} \) and where \( v_j = 1 \) and \( v_j = 0 \) characterize worker \( j \) as honest or adversarial, respectively. Accordingly, we rewrite the optimization problem as:

\[
\max_{\hat{Z}, w, v} \psi_{bp}(w, \hat{Z}, v) = \sum_{j=1}^{N} \sum_{i=1}^{T} \sum_{m=1}^{K} (v_j \hat{z}_{im} w_j \delta(r_{ij} - m)) \\
+ \frac{1}{K-1} (1 - v_j) \hat{z}_{im} w_j (1 - \delta(r_{ij} - m))
\]  

subject to

\[
\sum_{j=1}^{N} w_j^2 = 1, \quad \hat{z}_{im} \in \{0, 1\}, \quad \sum_{m=1}^{K} \hat{z}_{im} = 1, \quad v_j \in \{0, 1\}.
\]

Here, when a worker is identified as adversarial, we allocate equal weight \( \frac{w_j}{K-1} \) to all the answers except the one the worker has chosen. A locally optimal algorithm, maximizing the objective \( \psi_{bp} \) starting from an initial weight vector \( w = \epsilon \mathbf{1} \), consists of the following three iterated steps:

**Step 1:** For fixed values of \( w \) and \( v \) and for each task \( i \), the optimal \( \hat{z}_i \), maximizing (4.19), is chosen as:

\[
\hat{z}_{im} = \begin{cases} 
1 & \text{if } \sum_{j: r_{ij} = m} v_j w_j + \frac{1}{K-1} \sum_{j: r_{ij} \neq m} (1 - v_j) w_j > \\
\sum_{j: r_{ij} = k} v_j w_j + \frac{1}{K-1} \sum_{j: r_{ij} \neq k} (1 - v_j) w_j \forall k \neq m & \\
0 & \text{otherwise}
\end{cases}
\]  

The weights across incorrect answers are normalized by \( \frac{1}{K-1} \) so that each worker’s contribution to a given task equals his weight \( w_j \).
**Step 2:** For fixed values of \( \hat{Z} \) and \( v \), the optimal \( w \), maximizing (4.19), is given by:

\[
w_k^* = \frac{T \sum_{i=1}^{T} \sum_{m=1}^{K} \hat{z}_{im} v_k \delta(r_{ik} - m) + \frac{(1-v_k)}{K-1} (1 - \delta(r_{ik} - m))}{\sqrt{N \left( T \sum_{i=1}^{T} \sum_{m=1}^{K} \hat{z}_{im} v_j \delta(r_{ij} - m) + \frac{(1-v_j)}{K-1} (1 - \delta(r_{ij} - m)) \right)}}^{1/2} \quad (4.21)
\]

**Step 3:** For fixed values of \( \hat{Z} \) and \( w \), the optimal \( v \), maximizing (4.19), is:

\[
v_j = \begin{cases} 
1 & \text{if } \sum_{i=1}^{T} \sum_{m=1}^{K} \hat{z}_{im} w_j \delta(r_{ij} - m) \geq \frac{1}{K-1} \hat{z}_{im} w_j (1 - \delta(r_{ij} - m)) \forall j \\
0 & \text{otherwise}
\end{cases} \quad (4.22)
\]

### 4.3.2.2 Negative Weights Signify Adversaries

In the previous algorithm, we can see that

i) An honest worker’s weight contributes to the objective function only if he votes with the (weighted) plurality.

ii) Binary parameters included to represent worker intent result in an additional optimization step. As will be seen in the experiments section, this additional step may be a source of (poor) local optima.

iii) The weights computed by (4.21) will always be non-negative, and so the full (real line) range of \( w_j \) is not being utilized.

Here, we propose an approach which remedies all three of these “issues” associated with the previous algorithm. First, let us note that negative weights can be used to signify adversarial workers, and treated accordingly. Thus rather than apportion \( \frac{w_j}{K-1} \) when \( v_j = 0 \), as in Section 4.3.2.1, we can equivalently apportion \( \frac{|w_j|}{K-1} \) when \( w_j < 0 \) (and, as we shall see shortly, thus avoid the need to explicitly introduce binary intention parameters). Second, to appreciate a possible alternative to i), suppose an influential nonadversarial worker (\( w_j \) large and positive) does not agree
with the (current) weighted plurality decision for a given task. Rather than not contributing to the plurality score and the objective function, this worker could subtract his weight from the weighted plurality score (in fact, from the scores of all answers with which he disagrees) to try to alter the weighted plurality decision. Interestingly, we can achieve both of these mechanisms, and also avoid including binary intention parameters, with the introduction of a single additional cost term, thus modifying the objective function from Section 4.3.2.1. Specifically, supposing that $\hat{Z}$ is the current estimate of ground truth answers, then worker $j$’s contribution to the objective function is now taken to be:

$$w_j \sum_{i=1}^{T} \sum_{m=1}^{K} \left( \hat{z}_{im} \delta(r_{ij} - m) - \frac{1}{K-1} \hat{z}_{im} (1 - \delta(r_{ij} - m)) \right)$$  \hspace{1cm} (4.23)

Let us consider the two cases $w_j < 0$ and $w_j > 0$. If $w_j < 0$, then in (4.23) the adversary’s weight magnitude is (fully) subtracted from the current plurality score if his answer agrees with that of the current plurality. On the other hand, if his answer disagrees with the current plurality, his weight magnitude is equally apportioned amongst the remaining answers. Thus, (4.23) accounts for adversarial workers precisely as we intend. Next, suppose that $w_j > 0$. We can see that (4.23) behaves as desired in this case as well, fully adding $w_j$ to the plurality score if he agrees with the plurality decision, and, if he disagrees with the plurality, subtracting $\frac{w_j}{K-1}$ from the scores of the answers with which he disagrees. Thus (4.23) accounts for both adversarial and non-adversarial workers precisely in the way we intend. Accordingly, we modify the objective function using (4.23) as the per-worker contribution:

$$\max_{\hat{Z}, w} \psi_{\text{neg}}(w, \hat{Z}) = \sum_{j=1}^{N} \left( w_j \sum_{i=1}^{T} \sum_{m=1}^{K} \left( \hat{z}_{im} \delta(r_{ij} - m) - \frac{1}{K-1} \hat{z}_{im} (1 - \delta(r_{ij} - m)) \right) \right)$$  \hspace{1cm} (4.24)

subject to

$$\sum_{j=1}^{N} w_j^2 = 1, \hspace{0.5cm} \hat{z}_{im} \in \{0, 1\}, \hspace{0.5cm} \sum_{m=1}^{K} \hat{z}_{im} = 1.$$
We further note that, supposing \( \hat{Z} \) are the ground truth answers, then the per worker term bracketed in (4.24) \[ \sum_{i=1}^{T} \sum_{m=1}^{K} (\hat{z}_{im}\delta(r_{ij}-m) - \frac{1}{K-1}\hat{z}_{im}(1-\delta(r_{ij}-m))) \]
for a spammer \( j \) goes to 0 as \( T \to \infty \). This follows from the weak law of large numbers and from our assumption that a spammer will randomly choose an answer with a uniform distribution on all possible choices. Consequently, assigning a non-zero weight to a spammer is clearly sub-optimal due to the energy constraint on the weight vector, \textit{i.e.}, to maximize (4.24), spammers will (asymptotic in \( T \)) be assigned zero weights. Thus (4.24) properly accounts for honest workers, malicious workers and spammers.

Our locally optimal algorithm for (4.24) consists of iteration of the following two steps, starting from the weight vector initialization:

**Step 1:** For a fixed \( w \) and for each task \( i \), choose \( \hat{z}_i \) as

\[
\hat{z}_{im} = \begin{cases} 
1 & \text{if } \sum_{j:r_{ij}=m} w_j - \frac{1}{K-1} \sum_{j:r_{ij}\neq m} w_j > \sum_{j:r_{ij}=k} w_j \forall k \neq m \\
0 & \text{otherwise}
\end{cases} \tag{4.25}
\]

**Step 2:** For fixed \( \hat{Z} \), the optimal \( w \) is:

\[
w^*_k = \frac{\sum_{i=1}^{T} \sum_{m=1}^{K} \hat{z}_{im}\delta(r_{ik}-m) - \frac{1}{K-1}(1-\delta(r_{ik}-m))}{\sqrt{\sum_{j=1}^{N} \left( \sum_{i=1}^{T} \sum_{m=1}^{K} \hat{z}_{im}\delta(r_{ij}-m) - \frac{1}{K-1}(1-\delta(r_{ij}-m)) \right)^2}}. \tag{4.26}
\]

### 4.3.3 Semisupervised Case

Note that for both the methods in Sections 4.3.2.1 and 4.3.2.2, it is quite straightforward to incorporate probe task supervision. This is achieved by slightly modifying step 1 in both cases. Specifically, when a task is probe, we simply fix the value of \( \hat{z}_i \) to the ground-truth value, rather than using the weighted plurality decision rule.
4.4 Experiments

Experiments were performed using synthetic data as well as data generated by a crowdsourced multiclass labeling task on Amazon MTurk. Additionally, for a number of UC Irvine domains, we generated a collection of heterogeneous classifiers to be used as a “simulated” crowd. We generated adversaries of the simple type in all our experiments. In addition to the proposed schemes, we also compared with: the Dawid and Skeene method [52], which uses a confusion matrix to account for worker reliabilities; simple (multiclass) plurality voting; and its semisupervised version, which exploits the probe tasks. Let us reiterate the different methods that we will apply and study in this section:

i) PLU: Simple plurality.

ii) SS-PLU: Semisupervised plurality.

iii) US-DS: Unsupervised Dawid and Skeene EM algorithm [52].

iv) US-SW: Unsupervised objective-based method that uses binary parameter for intention described in Section 4.3.2.1.


vi) US-NEG: Unsupervised objective-based method without intention parameters, described in Section 4.3.2.2.


viii) SS-GEM: Semisupervised GEM based scheme described in Section 4.2.


4.4.1 Experiments with Synthetic Data

These experiments were performed in two parts. For the first part, the synthetic data was produced according to the stochastic generation described in Sections

---

This is a weighted plurality scheme, where each worker is weighted by the fraction of probe tasks that he answered correctly.
4.2.2 and 4.2.3. The goal here was to evaluate the GEM algorithm by comparing the estimated parameters and the estimated hidden ground truth answers with their actual values used in generation of the synthetic data. We generated a crowd of 100 workers with $d_j \sim \mathcal{N}(1, 400)$, $a_j \sim \mathcal{N}(0.3, 0.2)$; 10% of workers were adversarial. The tasks were generated with $\tilde{d}_i \sim \mathcal{N}(8, \sigma^2)$, where $\sigma^2$ was varied. The ground truth answer for each task was chosen randomly from $\{0, 1, \ldots, 4\}$. We observed that in this regime of high variance in the generated values of worker skill and task difficulty, there is a definite advantage in using the GEM based schemes (SS-GEM and US-GEM) over other schemes, as shown in Table 4.1. We also see in Figure 4.1 the high correlation between the estimated and actual values of worker skills and task difficulties. Also noteworthy from Table 4.1 is the superior performance of US-GEM over the other unsupervised objective-based schemes such as US-SW and US-NEG. Table 4.3 shows performance as a function of the number of workers assigned to each task. In each case, a random regular bipartite graph of workers and tasks was generated. We also observed that US-DS is highly sub-optimal in this high variance regime. We plot in Figure 4.2 the histogram of worker accuracies for one of the trials for the first entry in Table 4.1. This illustrates that highly skilled workers represent a small minority in these experiments. Next, we performed a comparative study for the “low variance” regime. Particularly, we generated the worker skill and task difficulty parameters as $d_j \sim \mathcal{N}(1, 1)$ and $\tilde{d}_i \sim \mathcal{N}(7, 1)$. We also introduced spammers (who randomly chose answers for all tasks with a uniform pmf over the answer space) whose proportion was varied. As seen in Table 4.2, the deterministic objective-based schemes are more suitable under this scenario. Figure 4.3 shows the histogram of worker accuracies in this regime. We can observe a clear difference between worker accuracy profiles in Figures 4.2 and 4.3. These results indicate that the GEM-based approach is apparently suitable when a small minority of highly skilled experts are present in a crowd of mostly low skilled workers with varying skill levels, whereas the objective-based approach seems to be ideal when a few skilled workers (not necessarily very highly skilled) are present in a crowd containing largely spammers and the variance in skill levels across the workers is low.

In the second part we separately evaluated the deterministic objective-based schemes (US-SW and US-NEG) and compared them with simple plurality voting.
We performed this for two different models of synthetically generated data. For the first model, all workers belonged to one of three categories: spammers, “hammers” and adversarial workers. For all tasks, the spammers answered randomly, the hammers were always correct, whereas the adversarial workers always chose randomly from the incorrect answers. Table 4.4 shows comparison across varying proportions of spammers and adversarial workers in the crowd. We can observe that US-DS outperforms PLU, but US-GEM and US-NEG give clearly superior performance. For the second model, we generated the data according to the stochastic model in Section 4.2.2 with the task difficulty for each task \( \tilde{d}_i \sim U[0,8] \). We created three categories of workers: high-skilled honest workers \( (d_j \sim U[0,8]) \), low-skilled honest workers \( (d_j \sim U[0,2]) \), and high-skilled simple adversarial workers \( (d_j \sim U[0,8]) \). The adversarial workers answered incorrectly to the best of their skill level according to (4.2). Table 4.5 shows the comparison of the schemes for this model of synthetic data. Note that for both the experiments in Tables 4.4 and 4.5, we averaged the number of erroneous tasks over 20 trials. We can observe from Tables 4.4 and 4.5 that US-NEG clearly outperforms US-SW and PLU.

One hypothesis for the performance advantage of US-NEG over US-SW is the additional “layer” of optimization needed to choose the binary intention parameters in US-SW. This could give a greater tendency of US-SW to find (relatively) poor local optimum solutions of its objective function. To test this hypothesis, we devised a hybrid method (US-HYB) which maximizes the US-SW objective function, but starting from \( \hat{Z} \) and \( \hat{v} \) initialized based on the US-NEG solution (with the sign of \( w_j \) used to initialize \( v_j \)). We can see from Table 4.4 that US-HYB certainly performs better than US-SW. Table 4.6 shows the number of trials (out of 100) when both i) US-HYB gave fewer decision errors than US-SW and ii) the objective function value using US-HYB was strictly greater than using US-SW. We can see from the entries in this table and Table 4.4 that there is a strong correlation between US-HYB achieving a greater US-SW objective function value and US-HYB achieving fewer decision errors than US-SW.
Figure 4.1: Comparison of actual and estimated parameters.

Figure 4.2: Histogram of worker accuracies for high variance regime.

Figure 4.3: Histogram of worker accuracies for low variance regime.
<table>
<thead>
<tr>
<th>Task Variance</th>
<th>Average erroneous tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>23.9</td>
</tr>
<tr>
<td>2000</td>
<td>21.7</td>
</tr>
<tr>
<td>1000</td>
<td>19.4</td>
</tr>
<tr>
<td>500</td>
<td>10.9</td>
</tr>
<tr>
<td>250</td>
<td>5.8</td>
</tr>
</tbody>
</table>

Table 4.1: Synthetic data generated using the stochastic model for the high variance regime: Changing task difficulty variance

<table>
<thead>
<tr>
<th>Percentage Spammers</th>
<th>Average erroneous tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>16.2</td>
</tr>
<tr>
<td>30</td>
<td>18.4</td>
</tr>
<tr>
<td>40</td>
<td>34.1</td>
</tr>
<tr>
<td>50</td>
<td>36.2</td>
</tr>
</tbody>
</table>

Table 4.2: Synthetic data generated using the stochastic model for the low variance regime: Changing proportion of spammers

### 4.4.2 Simulating a Crowd using an Ensemble of Classifiers

We also leveraged ensemble classification to generate a set of automated workers (each an ensemble classifier) using boosting [61]. Each such classifier (worker) is a strong learner obtained by applying multiclass boosting to boost decision tree-based weak learners. The strength (accuracy) of each worker was varied by controlling the number of boosting stages. Each weak learner was trained using a random subset of the training data to add more heterogeneity across the workers’ hypotheses. Note that unlike Section 5.1, this approach to simulated generation of a crowd is not obviously matched to our stochastic data generation model in Section 4.2.2. Thus, this more complex simulation setting provides a more realistic challenge for our model and learning.

We ran Multiboost [62] 100 times to create a crowd of 100 workers for four domains that are realistic as crowdsourcing domains: Pen Digits, Vowel, Dermatology, and Nominal. For each experimental trial, 100 crowd tasks were created.

---

We resampled the dataset to have only odd digits.

Hungarian named entity dataset [63]. Identifying and classifying proper nouns into four categories: not a proper noun, person, place, and organization.
## Assignment degree

<table>
<thead>
<tr>
<th>Assignment degree</th>
<th>Average erroneous tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>31.4</td>
</tr>
<tr>
<td>40</td>
<td>22.6</td>
</tr>
<tr>
<td>60</td>
<td>19.9</td>
</tr>
<tr>
<td>80</td>
<td>16.3</td>
</tr>
</tbody>
</table>

Table 4.3: Synthetic data generated using the stochastic model for the high variance regime: Changing number of worker assignments.

## Percentage adversaries

<table>
<thead>
<tr>
<th>Percentage adversaries</th>
<th>Number of erroneous tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Percentage spammers</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>5.4/5.1/5.6/5.8/6.0</td>
</tr>
<tr>
<td>5</td>
<td>5.6/5.4/5.3/5.4/5.5</td>
</tr>
<tr>
<td>10</td>
<td>6.8/6.1/6.0/5.5/5.0</td>
</tr>
<tr>
<td>15</td>
<td>5.8/5.0/5.0/5.0/5.0</td>
</tr>
<tr>
<td>20</td>
<td>8.1/7.1/7.0/6.0/5.0</td>
</tr>
</tbody>
</table>

Table 4.4: Synthetic data: Comparing objective-based methods for spammer, hammer and adversary model.

by randomly choosing 100 data samples from a given domain; 10 of them were randomly chosen to be probe. The rest of the data samples from the domain were used for training the strong (ensemble) classifiers/workers. The average of the number of crowd-aggregated erroneous tasks was computed across 5 trials, where each trial consisted of a freshly generated crowd of workers and set of tasks. In Tables 4.7, 4.8, 4.9, and 4.10 we give performance evaluation for different worker accuracy means and variances for the four domains. We did not directly control these values since they were an outcome of the boosting mechanism. However we could control the number of boosting stages used by each worker. We also show the performance when 10% of the workers were replaced by adversarial workers. These synthetic adversaries retained the estimated skill level of the (replaced) workers.
Table 4.5: Synthetic data: Comparing objective-based methods for data generated using model in Section 4.2.2

<table>
<thead>
<tr>
<th>Percentage high skilled adversarial workers</th>
<th>Number of erroneous tasks PLU/US-SW/ US-NEG/US-DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>9.4/6.7/, 6.0/9.6, 10.5/6.5/, 6.1/10.4, 12.7/6.9/, 6.7/12.5, 14.4/8.3/, 7.2/14.2, 20.3/11.3/, 10.5/19.4, 26.9/17.3/, 15.4/23.6</td>
</tr>
</tbody>
</table>

Table 4.6: Comparing US-SW and US-HYB

<table>
<thead>
<tr>
<th>Percentage Adversarial</th>
<th>Number of trials Percentage Spammers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>24</td>
</tr>
<tr>
<td>15</td>
<td>71</td>
</tr>
<tr>
<td>20</td>
<td>55</td>
</tr>
</tbody>
</table>

and generated their answers using the stochastic model described in section 3.2. In Table 4.7, the worker accuracy mean and variance across all workers is based on the number of correctly answered tasks (both probe and non-probe) for each worker computed in the absence of adversaries.

We can see in Table 4.7 for the Pen Digits dataset, the gain in performance with our methods, especially in the presence of adversarial workers. Note that for low means of worker accuracy, the GEM based methods outperform others, whereas the weighted plurality based methods using negative weights for adversaries (US-NEG and SS-NEG) perform better than other schemes for relatively higher means of worker accuracy. Figures 4.4 and 4.5 show the histogram of worker accuracies for low and high mean cases, respectively. In Figure 4.4, we plot the histogram
of worker accuracies corresponding to the first entry in Table 4.7 (for a single trial). We can observe an extremely skewed distribution of worker accuracies where a tiny minority of extremely high skilled workers exist amidst others who are mostly spammers. This distribution seems to best “agree” with the GEM based schemes. Figure 4.5 corresponds to the last entry in Table 4.7. Here we observe a less skewed distribution of worker accuracies. The objective based US-NEG and SS-NEG schemes perform better in this regime. The GEM based scheme is able to exploit simple adversaries much more than other methods, achieving improved crowd aggregation accuracy compared to the case where no adversaries are present, for almost all the entries in Table 4.7. Also observed from the table on this dataset, 10% probe task supervision does not greatly improve performance, e.g., US-GEM performs as well as SS-GEM. Moreover, SS-PLU is highly sub-optimal when probe tasks are available. From these results, we can see that our proposed schemes are able to identify and leverage the expertise of a small subset of highly skilled workers, thus defeating “tyranny of the masses”. As seen in Table 4.8 on Dermatology, although unsupervised, US-NEG performs very close to SS-GEM when adversarial workers are absent. However, its performance degrades when adversaries are introduced in the crowd. SS-GEM greatly outperforms other methods. Also note that, on this dataset, unlike Pen Digits, probe supervision greatly assists the inference using GEM for lower mean worker accuracy (as seen in the first two entries of Table 4.8). In Figures 4.6 and 4.7, we plot the worker accuracy profiles for two cases: where probe supervision is and is not greatly beneficial, respectively. Table 4.9 shows the results on the vowel dataset. Here, US-NEG and SS-NEG clearly outperform all other methods, even SS-GEM, in the absence of adversaries. US-GEM and SS-GEM, again, perform better than all others when adversarial workers are introduced. Figure 4.8 shows the distribution of worker accuracies in the crowd in a histogram for one of the trials corresponding to the lowest mean accuracy in Table 4.9. We observed from our experiments with the Nominal dataset (Table 4.10) that the GEM based methods perform better than others overall, although the performance gap diminishes as the mean worker accuracy improves.
### Table 4.7: Experiments using Pen Digits dataset: Average number of erroneous tasks

<table>
<thead>
<tr>
<th>Worker accuracy mean</th>
<th>Worker accuracy variance</th>
<th>Task accuracy variance</th>
<th>Without adversarial workers</th>
<th>With 10% adversarial workers</th>
</tr>
</thead>
<tbody>
<tr>
<td>24.73</td>
<td>271.3</td>
<td>26.16</td>
<td>23.5</td>
<td>10.76</td>
</tr>
<tr>
<td>27.01</td>
<td>305.2</td>
<td>35.2</td>
<td>21.8</td>
<td>10.6</td>
</tr>
<tr>
<td>30.15</td>
<td>472.3</td>
<td>51.9</td>
<td>17.5</td>
<td>11.4</td>
</tr>
<tr>
<td>34.2</td>
<td>500.2</td>
<td>90.1</td>
<td>14.6</td>
<td>8.4</td>
</tr>
<tr>
<td>39.7</td>
<td>534.6</td>
<td>193.1</td>
<td>13.1</td>
<td>7.2</td>
</tr>
</tbody>
</table>

4.4.3 MTurk Experiment

We designed an image labeling task where workers had to provide the country of origin, choosing from *Afghanistan, India, Iran, Pakistan, Tajikistan*. Some of the regions in these countries look very similar in their culture, geography, and demography and hence only people with domain experience and a deep understanding of the region will likely know the true answer. For instance, the blue rickshaws are typical to Pakistan and the yellow taxis are more common in Kabul. One can also guess *e.g.* from the car models on the street or from the script on street banners. We posted a task consisting of 50 such images on Amazon MTurk and asked all workers to upload a file with their answers on all tasks. We received responses from 62 workers. In order to evaluate under the scenario where workers
Figure 4.4: Pen Digits dataset: Histogram of worker accuracies with a skewed distribution of skills

Figure 4.5: Pen Digits dataset: Histogram of worker accuracies with a more gradual “spread” of skill distribution
Figure 4.6: Dermatology dataset: Histogram of worker accuracies. Supervision greatly helps.

Figure 4.7: Dermatology dataset: Histogram of worker accuracies. Supervision does not help much.
<table>
<thead>
<tr>
<th>Worker accuracy mean</th>
<th>Worker accuracy variance</th>
<th>Task accuracy variance</th>
<th>Without adversarial workers</th>
<th>With 10% adversarial workers</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.9</td>
<td>225.2</td>
<td>48.2</td>
<td>46.1</td>
<td>22.9</td>
</tr>
<tr>
<td>23.3</td>
<td>405.6</td>
<td>72.4</td>
<td>39.7</td>
<td>11.8</td>
</tr>
<tr>
<td>27.4</td>
<td>500.1</td>
<td>78.9</td>
<td>22.1</td>
<td>7.9</td>
</tr>
<tr>
<td>29.1</td>
<td>629.1</td>
<td>95.4</td>
<td>25.6</td>
<td>8.7</td>
</tr>
<tr>
<td>33.1</td>
<td>652.4</td>
<td>120.1</td>
<td>12.2</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Table 4.8: Experiments using Dermatology dataset: Average number of erroneous tasks

<table>
<thead>
<tr>
<th>Worker accuracy mean</th>
<th>Worker accuracy variance</th>
<th>Task accuracy variance</th>
<th>Without adversarial workers</th>
<th>With 10% adversarial workers</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.8</td>
<td>381.7</td>
<td>32.3</td>
<td>26.2</td>
<td>16.1</td>
</tr>
<tr>
<td>26.2</td>
<td>405.3</td>
<td>46.5</td>
<td>23.8</td>
<td>14.5</td>
</tr>
<tr>
<td>28.4</td>
<td>434.3</td>
<td>54.6</td>
<td>22.5</td>
<td>13.9</td>
</tr>
<tr>
<td>31.8</td>
<td>618.4</td>
<td>78.4</td>
<td>19.1</td>
<td>12.6</td>
</tr>
<tr>
<td>36.2</td>
<td>732.2</td>
<td>124.4</td>
<td>16.8</td>
<td>13.9</td>
</tr>
</tbody>
</table>

Table 4.9: Experiments using Vowel dataset: Average number of erroneous tasks

answer only a subset of the tasks, for each task, we used answers from a sampled set of workers using a randomly generated degree regular bipartite graph consisting of worker and task nodes. A worker’s answer to a task was used only when a link existed in the bipartite graph between the two corresponding nodes. Table 4.11 shows the average number of erroneous crowd-aggregated answers for the
methods under study as we varied the number of tasks assigned to each worker. The average was computed over 5 trials, each time generating a new instance of a random bipartite graph and using 5 randomly chosen probe tasks. The histogram of worker accuracies is shown in Figure 4.10. From the histogram we can observe that only a small fraction of workers could answer most tasks correctly.

<table>
<thead>
<tr>
<th>Worker accuracy mean</th>
<th>Worker accuracy variance</th>
<th>Task accuracy variance</th>
<th>Without adversarial workers</th>
<th>With 10% adversarial workers</th>
</tr>
</thead>
<tbody>
<tr>
<td>44.2</td>
<td>562.3</td>
<td>341.5</td>
<td>20.4</td>
<td>18.2</td>
</tr>
<tr>
<td>48.8</td>
<td>604.5</td>
<td>331.9</td>
<td>10.5</td>
<td>8.5</td>
</tr>
<tr>
<td>50.8</td>
<td>642.6</td>
<td>326.6.1</td>
<td>9.9</td>
<td>7.9</td>
</tr>
<tr>
<td>52.9</td>
<td>682.5</td>
<td>254.6</td>
<td>9.5</td>
<td>6.1</td>
</tr>
</tbody>
</table>

Table 4.10: Experiments using Nominal dataset: Average number of erroneous tasks

Figure 4.8: Vowel dataset: Histogram of worker accuracies.
4.5 Related Work

Stochastic models for the generation of workers’ answers have been previously considered in [52], [64], [58], [48]. In [52] the parameters of the model are the per-worker confusion matrices that are jointly estimated with the ground truth answers. This method was extended in [65], where a distinct probability distribution was given over the answers for every task-worker pair. But [65] does not consider a per-task parameter representing task difficulty and the minimax entropy-based approach necessitates regularization using many hyper-parameters which need to be chosen via methods such as cross-validation. In our model, we do not require any hyper-parameters. Moreover, we explicitly model the interplay between worker skill and task difficulty.

Our focus has been mostly on crowdsourcing tasks that require an expertise (skill) present only in the minority of the crowd. In [58] and [48], task difficulty was considered explicitly, but only for the binary case. A multicategory extension
of the binary model in [58] was proposed by the authors in [66]. However it
does not well-model a worker’s response in a multicategory setting. Particularly,
in the model proposed in [66], an honest worker answers an impossible task (a
task with infinite difficulty parameter) with a probability equal to $\frac{1}{2}$, instead of
$\frac{1}{\text{NUMBER OF CHOICES}}$, which does not seem intuitive. This has also been pointed
out elsewhere [67]. [67] considered task difficulty for ordinal data, wherein a higher
difficulty level adds more variance to the distribution of workers’ elicited answers
without affecting the mean (which equals the ground truth value). Our method of
incorporating task difficulties is novel, as we use them in a comparative paradigm
with worker skill in our soft threshold-based model. Unlike [48], which assumes all
tasks are drawn from the same classification domain, our approach is applicable
even when the batch of tasks is heterogeneous, i.e., not necessarily all drawn from
the same (classification) domain. This is due to the fact that our model does not
consider domain-dependent features [48] and also because our model is invariant
to task-dependent permutations applied to the answer space. Note that [55] allows
biasing towards certain answers, which only applies when all the tasks belong to
the same classification domain.

Adversarial workers in the binary case were accounted for in [58] and [53]. In
this work, we characterized adversarial behavior for a more generalized (multicat-
egory) setting and proposed several realistic adversarial models. We also showed
how we can retain the interpretation of negative weights as representing adversaries
in generality from the binary [53] to the multicategory case. Moreover, we showed
that our approach exploits responses from (simple) adversaries to actually improve
the overall performance. [53] and [68] consider other statistical methods, such as
correlation-based rules and low rank approximation of matrices. These methods
have been studied for binary classification tasks. Our objective-based approach

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>15.4</td>
<td>18.2</td>
<td>15.6</td>
<td>15.1</td>
<td>15.1</td>
<td>14.3</td>
<td>14.3</td>
<td>13.9</td>
<td>13.4</td>
</tr>
<tr>
<td>20</td>
<td>14.2</td>
<td>13.6</td>
<td>14.8</td>
<td>14.1</td>
<td>14.1</td>
<td>13.8</td>
<td>13.7</td>
<td>7.2</td>
<td>7.2</td>
</tr>
<tr>
<td>30</td>
<td>14.2</td>
<td>11.2</td>
<td>14.4</td>
<td>13.5</td>
<td>13.4</td>
<td>13.2</td>
<td>13.2</td>
<td>5.2</td>
<td>5.2</td>
</tr>
<tr>
<td>40</td>
<td>12.2</td>
<td>7.6</td>
<td>13.1</td>
<td>11.5</td>
<td>11.5</td>
<td>10.1</td>
<td>10.1</td>
<td>4.2</td>
<td>4.0</td>
</tr>
<tr>
<td>50</td>
<td>13.4</td>
<td>8.0</td>
<td>13.9</td>
<td>13.2</td>
<td>13.2</td>
<td>12.5</td>
<td>12.5</td>
<td>3.4</td>
<td>3.4</td>
</tr>
</tbody>
</table>

Table 4.11: MTurk Experiment: Average number of erroneous tasks
generalizes the weighted majority theme of these papers to a multicategory case, incorporating honest workers, adversaries, and spammers. We note that recently, [69] have tried to extend the low rank approximation approach to the case when the tasks do not have a ground truth answer and the answers (from a categorical space) can be subjective. In this case, “schools of thought” are discovered via clustering and the average size of clusters for each task is representative of its ease (clarity).

4.6 Future Work Extensions

Our simple adversarial model accounts for workers who are “naively” malicious. Here, we present a model of a more strategic adversarial worker. Such a worker answers correctly for simpler tasks with difficulty level below a certain value. Assume \( \theta_j < d_j \) to be such a threshold for worker \( j \). The pmf \( \beta \) for this (complex) adversarial worker is given by:

\[
\beta(r_{ij} = l|A_{ij}, v_j = 0, z_i) = \begin{cases} 
\frac{1}{K-1} \left( \frac{1}{1+e^{-b_j(\theta_j-d_i)}} \right) \left( \frac{1}{1+e^{-a_j(d_j-d_i)}} \right) + \\
\frac{1}{K} \left( \frac{1}{1+e^{-b_j(\theta_j-d_i)}} \right) \left( \frac{1}{1+e^{-a_j(d_j-d_i)}} \right) + \\
\frac{1}{K} \left( \frac{1}{1+e^{-b_j(\theta_j-d_i)}} \right) \left( \frac{1}{1+e^{-a_j(d_j-d_i)}} \right) + \\
if l = z_i \\
otherwise 
\end{cases} 
\]

(4.27)

Here, essentially, the worker answers correctly with high probability for easy tasks \((\theta_j > \tilde{d}_i)\), he excludes the correct answer for more difficult tasks below his skill level, and for even more difficult tasks that defeat his skill level \((d_j < \tilde{d}_i)\), he answers correctly at random \((\frac{1}{K})\).

As the popularity of crowdsourcing systems grows, we will have to account for various types of malicious workers in the aggregation rule. In the future, we would like to comprehensively evaluate more complex adversarial models, including the one proposed here. We would also like to explore the possibility of different attacks on crowdsourcing systems, for instance collusion attacks, where a group of adversarial workers collude and submit the same (but incorrect) answer for a task. This may severely affect the performance of unsupervised methods, depending on
the size of the group. Although our methods apply to a batch of heterogeneous tasks (without domain annotation), a comprehensive numerical study is needed to evaluate the comparative performance gain using our method in this setting. We can also consider the case where there is such domain annotation. In this case, a worker can have separate skill parameters for each of the domains present in the batch. We will also study the relationship between our objective-based schemes and low rank approximation schemes [68] and attempt to extend message passing schemes [53] to a multicategory setting.

Due to the non-convexity of the underlying optimization objective functions (the energy constrained weighted plurality function for the objective-based approach as well as the incomplete data log likelihood function for the generative model-based approach), we can guarantee only locally optimal solutions. However, based on our experiments, we observed that an unbiased initialization of parameters, *i.e.*, setting the same initial value of parameters across all workers and tasks, provides good, robust estimation accuracy in the case of synthetic data. Moreover, our approaches gave superior performance to the comparison methods irrespective of local optima issues. We will, in future, also try to address this issue in detail by evaluating some meta-heuristic approaches to overcome locally optimal solutions.
Chapter 5

Sybil Detection by Leveraging Properties of Trust Graphs

Reputation systems depend largely on the notion of unique user per “identity” i.e., each entity or an account is “owned” by a single user. In situations where creating identities is cheap and easy, a user can maliciously derive undeserving benefits from a reputation system by creating multiple identities of himself. This is popularly known as a “sybil attack” [50]. The benefits of creating multiple identities could be several. In the case of peer-to-peer (P2P) and other decentralized ad-hoc systems, newly joined peers may be provided with benefits such as a suitable starting reputation score that allows them to be “accepted” in the system. In this case, each new identity amounts to whitewashing a previous low reputation score and starting fresh. Also, a user might create fake identities that highly recommend each other and mislead honest users into believing that they have high reputation scores. Such an attack is typical to a system employing a distributed reputation system such as Eigentrust [70]. Solutions to sybil attacks use trusted certification, recurring costs for identity acquisition, resource testing and offline trust, among other techniques, to detect or penalize multiple identities created by single individual.
5.1 Reputation Systems and Trust Graphs

Multi-agent systems are vulnerable to several different forms of attacks, where agents may resort to malpractice and malicious activity, undermining the communal goals of the system, for example, fair distribution of content in a P2P system. Reputation systems are used to penalize such behavior and encourage agents away from such behavior. They are based on a metric of reputation score which is an indication of the past behavior of an entity. The methods and factors used in computing a reputation score can vary between systems and a universally accepted reputation metric is difficult to formulate because it should be truly representative of the entity’s behavior depending on the context. In the case of a completely decentralized system, a reputation score is relative and many different methods have been proposed for achieving consensus among agents on a commonly observed reputation score. A distributed reputation system ordinarily provides:

1. A function that maps transaction outcome to reputation
3. A function that maps the reputation of an entity requesting service to the level of service it should get.

The level of satisfaction from a transaction is relative to the entities involved, which, in turn means that the trust level enjoyed by an entity is different in the eyes of different entities. Some authors, e.g. those of [70], define reputation as the aggregation of trust in the eyes of all other entities. A trust graph is a directed graph whose nodes represent the interacting entities and the edges represent the “formal” relationships they share. The edge weights are the trust values and node weights their reputation. In some papers such as [71] papers the edges of trust graphs might represent “informal” or offline relationships.
5.1.1 Hierarchical Distributed Algorithm for Sybil Detection

5.2 Dense Connectivity Assumption

We assume that the graph of direct reputation, linking a cluster of honest nodes, has dense connectivity or a high value of sparse cut (defined below). This implies that a substantially large cluster of connected nodes can be separated from the rest of the graph by a proportionally large number of edges. This property fails when sybil nodes are present because a direct reputation link with an honest node is difficult to establish compared to a referral link. Hence a sybil cluster is typically sparsely connected via direct reputation links with the rest of the reputation graph [71]. And so identifying a sparse cut in a reputation graph will help us to identify sybil nodes. Our assumption that reputation graphs consisting of only honest nodes will be densely connected is based on [72], which states that dense connectivity can be a consequence of the presence of long-range links in graphs. In on-line peer-to-peer (P2P) social networks, if peer nodes have a propensity to establish new trust links with nodes other than those close to their own neighbors, the trust graph will be densely connected. Also, dense connectivity can be a consequence of the existence of a sufficiently dense set of highly connected super-peers or hubs.

5.3 Network Model and Sparse Cuts

Consider a connected graph $G = (V, E)$ where $V$ is the set of peers and $E$ is the set of edges representing direct reputation links between them. For the moment we will ignore the relationships due to referral paths. The nodes and the links in the graph are weighted, and the weight of a link is denoted by $w(u, v)$, where $u, v \in V$. Due to the asymmetric nature of trust relationships, $w(u, v) \neq w(v, u)$ generally. The weight of a cluster $S$, denoted $|S|$, is the combined weight of all the nodes in the cluster. Suppose we cut (partition) the graph $G$ into two sub-graphs with $A$ and $A'$ representing the two sets of connected nodes in the two partitions.
The sparsity of this cut is given by:

\[
\tilde{C}(A) = \frac{\sum_{(u,v): u \in A, v \in A'} w(u, v)}{\min(|A|, |A'|)}
\]

where \(w(u, v) = 0\) if \((u, v) \notin E\). This definition is similar to the Cheeger Isoperimetric Constant in Riemannian geometry [73]. We can slightly modify the above definition of sparsity of a cut in order to accommodate the asymmetric nature of trust relationships:

\[
C(A) = \min \left( \frac{\sum_{(u,v): u \in A, v \in A'} w(u, v)}{|A'|}, \frac{\sum_{(v,u): v \in A', u \in A} w(v, u)}{|A|} \right) \tag{5.1}
\]

Note that \(C(A) = C(A')\). The sparsity of the graph \(G\) is the minimum value of sparsity considering all the different ways in which we can partition the graph:

\[
S(G) = \min_{A \subseteq G} C(A) \tag{5.2}
\]

Figure 5.1: A sparse cut

5.4 Police Nodes

The presence of trusted nodes or super peers in the reputation graph is required in most existing referral frameworks. A verifier node which is honest is implicitly
assumed in [71] and [74]. According to [75], the presence of trusted nodes makes the reputation system asymmetric and hence it can be made sybil-proof. In our graph, we assume the presence of trusted nodes which we call police nodes. These could be specially embedded nodes purely for the purpose of sybil defense or nodes which have a long history of being trustworthy and helpful. A police node keeps track of trust relations between mostly regular nodes in its close vicinity of the trust graph. Hence it has a limited local view of the trust graph. It can acquire the information about the trust links from other nodes in its vicinity. It can also infer trust relationships by monitoring traffic it relays. We assume that police nodes are dense in the graph and well connected to each other by short geodesic (hop) distances, and every regular node is assumed to be only a short geodesic distance to some police node. Hence, police nodes occupy strategically important positions in the graph handling relatively heavy traffic loads, and can make rough inferences of social trust between its neighboring nodes by, e.g., simply monitoring their associated traffic volumes. Alternatively, regular peer nodes may communicate their direct trust vectors to their local police node for purposes of referral. A police node can use such data to create a complete picture of its local trust graph and check it for consistency.

5.4.1 Dealing with Evasive Sybils

There are two ways in which a sybil node might try to mislead the police nodes: by lying about an attack edge or by lying about a sybil edge. The former case is easier to detect. The police node may request for reputation information and referral paths from each node in its vicinity and verify the consistency of the information provided by one node with that provided by the other, i.e., through some kind of (centralized) consensus procedure. Since one of the nodes attached to an attack edge is an honest node by definition, the sybil node cannot mask or forge an attack edge. If the honest node attached to the attack edge is beyond the view of the police node then the lie will be detected at the next hierarchy of the algorithm when neighboring police nodes exchange information with each other. Detecting a lie about a sybil edge is trickier and we need to assume additional capabilities for the police nodes to monitor traffic. If there exists at least one
honest node within the view of the police node which has been referred to one of
the sybil nodes using a sybil edge, then a sybil node cannot lie about it, since the
honest node will enlighten the police node about its presence. Here we assume
that a sybil node will always try to mask a sybil edge to generate a fake sparse cut
within the sybil cluster. On the other hand, forging a non-existing sybil edge will
only harm the attacker by making the sybil cluster connect densely within itself
and hence making the attack edges appear sparser thereby improving detection
probability. If such an honest node does not exist, then the police node needs to
rely on the trust relationships it estimates by monitoring traffic in its vicinity as
described above.

5.5 Background of the Algorithm

5.5.1 The Hierarchical Framework

The aim of the following algorithm is to identify sybil clusters from a decentral-
ized database of reputation/referral paths in a distributed fashion with minimum
inter-node communication. It’s a scalable and computationally efficient alternative
compared to an approach involving data aggregation and centralized decision mak-
ing. The algorithm works at incremental hierarchical levels, where at each level
the following steps are taken:

1. Neighborhood discovery (and trust-links monitoring for the first hierarchy)

2. Randomized clustering

3. Determining sparse cut within each cluster if such a cut exists.

4. Collapsing each partition (or the whole cluster if a sparse cut is not detected)
to a single node

5. Incrementing the hierarchy level index

Incrementing the hierarchy implies considering the collapsed partition as a single
node and the aggregated inter-partition edges as inter node edges. After collapsing
the two partitions in a cluster, the police node in charge of the cluster represents
both the nodes formed after the collapse.
5.5.2 Penalty for Hierarchical Sparse Cut Computation

Hierarchical sparse cut computation is clearly different from a centralized approach and might incur a penalty (typically an underestimation error) in terms of the sparse cut identified. This is due to the limited view that the nodes have and that cuts in the collapsed nodes may not share common end points for splicing. However, even when the sparse cut detection algorithm is centralized, heuristic methods are often employed. Our focus here is not a precise calculation for the whole graph, rather a scalable approximation that is useful for identifying sybil identities. The final outcome of our algorithm depends on the clustering performed at each hierarchy. For this reason, we randomized clustering in our algorithm when evaluating its performance.

5.6 The Algorithm

5.6.1 Randomized Clustering

The algorithm to find a sparse cut in a graph is a heuristic which might not find a sparse cut even after multiple trials. However, the probability of identifying such sparse cut is quite good. The cluster heads wait for a random time before contacting nodes in successive hops. Note that our clustering objective is not to simply group similar objects together, rather we are interested in identifying sparse cuts within clusters that we suspected to be attack edges. In randomized clustering, each node randomly decides to become a cluster head randomly (except at the first hierarchy where all the police nodes are the cluster heads). Each cluster head tries to rope in nodes in its vicinity in successive hops starting with directly connected one hop nodes. A police node queries a node it wishes to rope-in to find whether it has already joined a different cluster. If so, the requested node replies to the requesting node with the id of its cluster head. In this way the cluster head learns about its neighboring cluster heads. The cluster head waits for a random amount of time before it contacts more distant nodes. The expected value of this random waiting time increases after each hop. This ensures that the cluster size does not become too large. The cluster head continues to expand its cluster until it gets negative responses from all the nodes at that distance. In order to limit
the size of the clusters at higher hierarchies each cluster head might keep a track of the number of nodes it has grouped so far. When it encounters another cluster, it might also compete with neighboring cluster head based upon the size of its cluster.

5.6.2 Sparse Cut Computation and Partition Collapse

Min-cut determination is a well known problem and is closely linked with finding maximum flow in a graph. The heuristic [76] is simpler than Ford-Fulkerson [77] and can be easily modified for our purposes to find a "sparse" cut. Each cluster-head tries to find a sparse cut within its cluster using the algorithm originally presented in [76], called there the contraction algorithm, where it is shown that by collapsing two nodes along a randomly selected edge until only two clusters remain, a min-cut with high probability will be found after $O(n^2)$ trials where $n$ is the number of vertices in the graph. Since the number of nodes in each cluster is limited in our case, the number of iterations required to find a sparse cut should be fairly small. For a sufficiently small cluster, we can also choose to do a brute-force cut search. If the sparseness of the min-cut is less than a threshold value then two partitions are formed within the cluster, else the complete cluster is considered as a single partition. Again, each element of the partition is collapsed to a single node for the next hierarchy and the police node represents the partitions formed from its cluster.

5.6.3 Incrementing Hierarchical Level

As nodes in a partition are collapsed into a single aggregate node, the new node thus formed has a weight equal to the sum of the weights of all aggregated nodes. We assign weight one to all the nodes at the first hierarchy. The weight of the edge between the new nodes formed in a given direction is equal to the combined weight of all the edges between the collapsed partitions in that direction. Maintaining node and edge weights this way allows sparse cuts to better approximate the corresponding operations on the lowest level graph. After incrementing the hierarchical level, the graph becomes coarser and we repeat all the steps beginning with randomized clustering at the new hierarchical level.
Figure 5.2: Hierarchical sparse cut detection algorithm: Starting with randomized clustering, inter cluster sparse cut detection, partition collapse, incrementing the hierarchy and randomized clustering at the next hierarchy.

5.7 Some Definitions and Analysis

An honest edge is one connecting two honest nodes or one from a sybil node to an honest node. A sybil edge is one connecting two sybil nodes. An attack edge is an edge from an honest node to a sybil node. In these definitions we assume the edge from a sybil node to an honest node to be honest because it represents the trust earned by the honest node in the eyes of the sybil node. A homogeneous partition is one containing all similar nodes (either all sybil nodes or all honest nodes). In a cluster of sybil nodes, we refer to those sybil nodes with attack edges directed towards them as legitimate sybil nodes. The sybil attacker can have more than one legitimate identity, i.e., more than one identity to which attack edges are directed. Sybil identities without attack edges are called pure.

If there is just one legitimate sybil identity, all sybil identities will belong to one cluster after the first iteration. This is because the cluster head that ropes in the first (legitimate) sybil will surely rope in the rest of the sybils (pure). Then, with
a high probability, they will be cleanly grouped by sparse cut into one aggregated node. That is, this aggregated node will contain all sybil identities and will be represented by the police node which was the cluster head of the original cluster. As we ascend the hierarchy, this aggregated node will not be further aggregated as this value of sparse cut will remain intact due to the absence of any other edge connecting this aggregated node with the rest of the graph so that, at the end, there will be just two clusters: the sybils and everything else. Hence the detection of a sybil with one legitimate sybil node is straightforward to analyze. It depends only on finding the sparse cut during the first hierarchy.

Now suppose there are multiple legitimate sybil identities of the sybil attacker. In the first iteration, it is possible that the legitimate sybil identities along with pure sybils will be partitioned and grouped into different clusters, where pure sybil identities can only be found together with at least one legitimate sybil identity. Nevertheless, sparse cuts will isolate the larger sybil clouds after the first iteration, and these larger sybil clouds will be grouped together in subsequent iterations. Due to our assumption that the sybil cloud is separated by a sparse cut, we can argue that the probability of an attack edge belonging to the sparse cut in a given hierarchy, provided that it belongs to the sparse cut in the previous hierarchy, increases with the level of the hierarchy. Similarly, we can argue using our assumption about the dense connectivity of networks containing honest nodes; that the probability of an honest edge belonging to the sparse cut in a given hierarchy, provided that it belongs to the sparse cut in the previous hierarchy, decreases with the level of the hierarchy.

As compared to the case of one legitimate sybil identity, the attacker has a better chance of evading detection if it distributes the attack edges among a larger set of legitimate sybil identities. Diluted reputations per legitimate identity will, however, naturally result for the same honest transactional effort. This is a fundamental trade-off when a defense based on sparse cuts is deployed.

We now provide a simple calculation for the case of a single sybil attacker with more than one legitimate identity. To this end, we first fix the following quantities:

- the total number of pure sybil identities, \( N_{ps} \)
- the mean edge-degree of an honest node, \( D \)
• the number of honest nodes in an iteration-1 cluster, \(N_h\), containing one legitimate sybil identity

• the number of attack edges, \(A\), of that legitimate sybil identity.

Let \(n_{ps}\) be the random number of pure sybil identities which are present in the iteration-1 cluster together with one legitimate sybil identity. Assume \(n_{ps}\) is uniformly distributed on \([0, N_{ps}]\), as would be the case if the sybil attacker has two legitimate identities in total. Assume the degrees of the honest nodes are independent as would be the case in a branching process approximation of the honest portion of cluster [78], beginning say at the legitimate sybil identity. Let \(d_{\text{min}}\) be distributed as the minimum degree among honest nodes in this cluster and assume that the degree distributions of the honest nodes are geometrically distributed so that

\[
E(d_{\text{min}}) = \frac{D}{N_h} \tag{5.3}
\]

**Theorem 4.** For this highly idealized model and for all sufficiently large \(A\),

\[
P\left(\frac{A}{n_{ps}} > d_{\text{min}}\right) > \frac{A/n_{ps}}{D/N_h} \tag{5.4}
\]

**Proof.**

\[
P\left(\frac{A}{n_{ps}} > d_{\text{min}}\right) = P(n_{ps} < \frac{A}{d_{\text{min}}})
\]

\[
= \sum_{i > 0}^{i < A} P(n_{ps} < \frac{A}{i})P(d_{\text{min}} = i)
\]

\[
= \frac{A}{N_{ps}} \sum_{i > 0}^{i < A} i^{-1}P(d_{\text{min}} = i) \quad \text{since } n_{ps} \sim U[0, N_{ps}]
\]

\[
= \frac{A}{N_{ps}} E\left( \frac{1}{d_{\text{min}}} \right) \quad \text{for all } A \text{ sufficiently large.}
\]

\[
> \frac{A/n_{ps}}{D/N_h} \quad \text{by Jensen’s Inequality.}
\]

\[\square\]
5.8 Simulation Study

Simulations were performed using NetLogo [9], a multi-agent programmable modeling environment. We programmed the behavior of a single node and observed the combined behavior of the network of nodes each operating concurrently. The trust graph simulated consisted of 225 honest nodes, each with degree with average 3 and varying from 1 to 6. An honest node connected to a distant honest node with positive probability, so that the graph became densely connected. Each honest edge connected an honest node with one of the other honest nodes selected randomly from a set of 50 closest honest nodes with a probability of 0.25. A sybil cluster of 25 nodes, which was densely connected within itself, was connected to this graph by a sparse number of attack edges. We randomly chose sybil nodes from among these to act as legitimate sybil nodes. The police nodes were randomly selected from the set if honest nodes. All the honest nodes were equally likely to be selected as police nodes in our simulations. The number of police nodes was fixed at 22. Note that mean cluster size is inversely proportional to the number of police nodes.

In the first iteration, these police nodes performed randomized clustering and roped in as many nodes in their vicinity as they could into their cluster. Each cluster head (police node) identified a sparse cut within its cluster using the contraction algorithm. Again, according to [76], \(O(n^2)\) trials are sufficient to determine a sparse cut. Hence we fixed the number of trials to 1000 given that the number of nodes in a cluster should include on average 10 honest nodes. We observed through our simulations that this value of number-of trials for the contraction algorithm, for determining the intra-cluster sparse cut, was indeed sufficient.

For the 250-node graph, a 3-level hierarchy was appropriate because clusters formed by a factor of 6 to 1 after the first iteration. Again, the outcome of the complete run was considered successful if, after the final hierarchical cut, all the sybil nodes were placed in one partition and all the honest nodes in the other.

Fig. 5.3 shows the average probability of detecting a cut in a single 3-level hierarchy trial plotted against the number of attack edges. We kept the number of legitimate sybil nodes constant at 2. We varied the number of attack edges from 2 to 8. Every attack edge connected an honest node to a sybil node which
was randomly chosen from the two legitimate sybil nodes. We ran an independent 3-level hierarchy trial of our algorithm for sybil detection on 300 different graphs generated randomly as described above. So, the sample standard deviation (i.e., statistical confidence) of the estimates is approximately \((p^2 - p^4)/300\)^0.5, where \(p\) is an average probability reported in Fig. 5.3. We observed the decrease in probability of success as the number of attack edges increased, which was expected since as the number of attack edges increase, the cut in the graph becomes less sparse. In the same experiment, we kept track of the probability of an event A when a single honest node is isolated, instead of the sybil nodes, in a cluster at the first level of the hierarchy. As discussed in the theorem above, we observed an increasing trend in this probability as we increased the number of attack edges.

Fig. 5.4 shows the result of a similar exercise where we instead varied the number of legitimate sybil nodes from 1 to 4. We kept the number of attack edges constant at 4. Every legitimate sybil node had at least one attack edge with an honest node. We ran a 3-level hierarchy trial of our algorithm on 100 different randomly generated graphs and found the average probability of detection. The results showed that the sybil attacker might improve its chances of evading detection by distributing the trust relationship it has with honest nodes among different sybil identities. But, again, this comes at the cost of diluting its accrued reputation as observed by honest nodes. Hence there is a tradeoff between distributing
Figure 5.4: The effect of number of legitimate sybil nodes on the detection probability. The number of attack edges is constant at 4.

In our next experiment we varied the small-world property of the trust graph in the absence of a sybil attacker. We did this by varying the probability of the attack edges between different legitimate sybil nodes to evade detection and the amount of reputation the legitimate sybil node can accrue for itself for transactional and referral benefit.

Figure 5.5: False positives for different sparsity threshold values on graphs with varying small-world property.

In our next experiment we varied the small-world property of the trust graph in the absence of a sybil attacker. We did this by varying the probability of
connecting to a distant node. As this probability was increased, the graph became more densely connected. We ran our sparse-cut detection algorithm on this graph and observed the percentage of false positives. As observed in Fig. 5.5, the design of the sparsity threshold value is critical in minimizing the false positives and its calibration should take into consideration the measure of the small-world property of the trust graph consisting of only honest nodes.

5.9 Future Work and Discussion

An important assumption in this work is that the trust graphs are expander graphs and so, naturally they are fast mixing. But, the properties of the trust graph are dependent on the distributed reputation system that is implemented. A class of papers on network formation games deals with the study of evolution of graphs and their properties that arise when the nodes choose to connect with other nodes based on maximization of some utility function. So, the macroscopic properties of the resulting graphs are studied for a given microscopic behavior at the level of individual nodes. In future, one can study the properties of trust graphs which arise when nodes are guided by some selfish motive, for example acquiring content with high bandwidth in case of peer-to-peer network. One can use this insight to develop reputation systems that result in trust graphs with some macroscopic properties sensitive to the presence of sybils.
Conclusion

The focus of this thesis was on tasking in multi-agent systems. We studied in particular three domain and application specific scenarios: two of which proposed task decomposition methods to achieve communal goals and one where we studied aggregation methods to combine the results of agent’s. In our proposed approaches, we employed a variety of tools such as incentive mechanisms using game theory, generative probabilistic modeling and graph theoretic algorithms.

A new criterion for super-peer based clustering in P2P networks was proposed that combines the advantages of semantic-similarity based clustering to optimize query search hop count along with load balancing across super-peers that optimizes the query resolution time and reduces query loss due to buffer overflows. The criterion was derived from local parameters, but nevertheless decisions based on it are proved (both mathematically and empirically) to descend in a global cost that is representative of the overall system objective. We also showed that descent is guaranteed under certain assumptions even when nodes take decisions asynchronously. We compared the scheme with an alternative scheme which is a direct outcome of solving a centralized cost function. We also modeled a P2P file sharing system based on a Gnutella-like system to test the proposed scheme with other simpler and benchmark schemes. The proposed scheme performed best when we considered the average query resolution time as the figure of merit for the system and also swiftly reached the steady state condition (i.e., no peer transfers)

In our crowd aggregation work, we focused on crowdsourcing tasks that require an expertise (skill) present only in the minority of the crowd. We proposed a novel
method of incorporating task difficulties, as we used a comparative paradigm between worker skills and task difficulties in our soft threshold-based model. Our approach is applicable even when the batch of tasks is heterogeneous, i.e., not necessarily all drawn from the same (classification) domain. This is due to the fact that our model does not consider domain-dependent features and also because our model is invariant to task-dependent permutations applied to the answer space. We also characterized adversarial behavior for a more generalized (multiclass) setting and proposed several realistic adversarial models. We showed how we can retain the interpretation of negative weights as representing adversaries in generality from the binary to the multiclass case. Moreover, we showed that our approach exploits responses from (simple) adversaries to actually improve the overall performance. Our objective-based approach generalizes the weighted majority theme to a multiclass case, incorporating honest workers, adversaries, and spammers.

We also proposed an agent-level hierarchical sparse-cut detection algorithm for distributed trust management and sybil detection. In particular, we considered the results of [71] in the light of direct and indirect reputations (referrals), defined a formal graph theoretic metric to describe the property of trust graphs sensitive to the presence of sybils. We proposed a distributed means of detecting sybil identity clusters by looking at local trust graphs (trust relationships in the close vicinity). The hierarchical approach was based on repeatedly finding sparse cuts in a localized cluster and successive coarsening of the graph making the algorithm hierarchical. We observed experimentally that the proposed approach is able to detect sybils with a high accuracy.
Appendix: Proof of Theorem 1 in Chapter 3

We can express $C_0(r)$ as the aggregation of four partial sums: i) cost of peer $l$ itself; ii) sum of costs of peers that belong to super-peer $k_1$ except $l$; iii) sum of costs of peers that belong to super-peer $k_2$ except $l$; and iv) sum of costs of those peers that belong neither to super-peer $k_1$ nor to super-peer $k_2$. In the three steps given below, we show that i), ii) and iii) decrease as peer $l$ changes its assignment to minimize its cost (peer-level) while iv) is left unchanged. Thus, $C_0(r)$ decreases, contradicting our assumption that $\hat{r}$ is the global optimum (minimum) solution of $C_0$. We can divide the objective function as follows:

$$C_0(r) = \left( \frac{b_l}{w_{r_l}} \sum_{j: r_j = r_l, j \neq l} b_j + \frac{\mu}{2} \sum_{j: r_j \neq r_l} c_{ij} \right) + \sum_{i: i \neq l, r_i = k_1} \left( \frac{b_i}{w_{r_i}} \sum_{j: r_j = r_i, j \neq i} b_j + \frac{\mu}{2} \sum_{j: r_j \neq r_i} c_{ij} \right)$$

$$+ \sum_{i: i \neq l, r_i = k_2} \left( \frac{b_i}{w_{r_i}} \sum_{j: r_j = r_i, j \neq i} b_j + \frac{\mu}{2} \sum_{j: r_j \neq r_i} c_{ij} \right) + \sum_{i: i \neq l, r_i \neq k_1, r_i \neq k_2} \left( \frac{b_i}{w_{r_i}} \sum_{j: r_j = r_i, j \neq i} b_j + \frac{\mu}{2} \sum_{j: r_j \neq r_i} c_{ij} \right)$$

Thus,

$$C_0(r) = C_l(r) + \sum_{i:i \neq l, r_i = k_1} C_i(r) + \sum_{i:i \neq l, r_i = k_2} C_i(r) + \sum_{i:i \neq l, r_i \neq k_1, r_i \neq k_2} C_i(r).$$
The first term is the cost of peer $l$, the second term is sum of the costs of the peers that are assigned to the former super-peer of peer $l$ (except peer $l$), the third term is the sum of the costs of the peers that belong to the prospective super-peer of peer $l$ (except peer $l$) and the fourth term is the sum of the costs of the peers that belong neither to the current super-peer $k_1$ of peer $l$ nor to the prospective super-peer $k_2$ of peer $l$.

Let the new assignment vector be $r^* = (r^*_1, r^*_2, ..., r^*_N)$, where $r^*_i = \hat{r}_i \ \forall \ i \neq l$.

$$C_0(r^*) - C_0(\hat{r}) = \left( C_l(r^*) - C_l(\hat{r}) \right) + \left( \sum_{i: i \neq l}^\prime C_i(r^*) - \sum_{i: i \neq l}^\prime C_i(\hat{r}) \right) + \left( \sum_{i: i \neq l, r^*_i = \hat{r}_i}^\prime C_i(r^*) - \sum_{i: i \neq l, r^*_i = \hat{r}_i}^\prime C_i(\hat{r}) \right).$$

**STEP 1:**

By assumption, we know that $C_l(r^*) - C_l(\hat{r}) < 0$.

**STEP 2:**

We can also deduce that

$$\sum_{i: i \neq l, r^*_i \neq \hat{r}_i} C_i(r^*) - \sum_{i: i \neq l, r^*_i = \hat{r}_i} C_i(\hat{r}) = 0,$$

because this term represents the sum of the change in the cost values of peers which belong to neither of the two super-peers involved in the transfer of peer $l$.

**STEP 3:**

Next we show that

$$\left( \sum_{i: i \neq l}^\prime C_i(r^*) - \sum_{i: i \neq l}^\prime C_i(\hat{r}) \right) + \left( \sum_{i: i \neq l}^\prime C_i(r^*) - \sum_{i: i \neq l}^\prime C_i(\hat{r}) \right) < 0.$$
First,

\[ \sum_{i \neq l, r_i^* = \hat{r}_i} \left( \frac{b_i}{w_{r_i^*}} \sum_{j: x_j^* = r_i^*} b_j + \frac{\mu}{2} \sum_{j: x_j^* \neq r_i^*} c_{ij} \right) - \sum_{i \neq l, \hat{r}_i = \hat{r}_l} \left( \frac{b_i}{w_{r_i}} \sum_{j: x_j^* = \hat{r}_l} b_j + \frac{\mu}{2} \sum_{j: x_j^* \neq \hat{r}_l} c_{ij} \right) \]

\[ = \sum_{i \neq l, r_i^* = \hat{r}_l} \left( \frac{b_i}{w_{r_i^*}} \sum_{j: x_j^* = r_i^*} b_j + \frac{\mu}{2} \sum_{j: x_j^* \neq r_i^*} c_{ij} \right) - \sum_{i \neq l, \hat{r}_i = \hat{r}_l} \left( \frac{b_i}{w_{r_i}} \sum_{j: x_j^* = \hat{r}_l} b_j + \frac{\mu}{2} \sum_{j: x_j^* \neq \hat{r}_l} c_{ij} \right). \]

Note that the two sets \( \{ i : i \neq l, \hat{r}_i = \hat{r}_l \} \) and \( \{ i : i \neq l, r_i^* = \hat{r}_l \} \) are equal because all other assignments except \( r_l \) are the same in the new assignment vector \( r^* \). Thus,

\[ \sum_{i \neq l} C_i(r^*) - \sum_{i \neq l} C_i(\hat{r}) = \sum_{i \neq l, r_i^* = \hat{r}_l} \left( \frac{b_i}{w_{r_i^*}} \sum_{j: x_j^* = r_i^*} b_j - \frac{b_i}{w_{r_i}} \sum_{j: x_j^* = \hat{r}_l} b_j + \frac{\mu}{2} \sum_{j: x_j^* \neq r_i^*} c_{ij} - \frac{\mu}{2} \sum_{j: x_j^* \neq \hat{r}_l} c_{ij} \right) \]

\[ = \sum_{i \neq l, r_i^* = \hat{r}_l} \left( \frac{b_i}{w_{r_i^*}} \sum_{j: x_j^* = r_i^*} b_j - \sum_{j: x_j^* = \hat{r}_l} b_j + \frac{\mu}{2} \left( \sum_{j: x_j^* \neq r_i^*} c_{ij} - \sum_{j: x_j^* \neq \hat{r}_l} c_{ij} \right) \right). \]

For \( i \neq l \), \( \{ j : r_j^* = \hat{r}_l, j \neq i \} \) represents the set of peers except the \( l \)th and \( i \)th peer which are currently assigned to \( \hat{r}_l \) super-peer and \( \{ j : \hat{r}_j = \hat{r}_l, j \neq i \} \) represents the set of peers except the \( l \)th and \( i \)th peer whose new assignment is the \( \hat{r}_l \)th super-peer. So, for \( i \neq l \), \( \{ j : r_j^* = \hat{r}_l, j \neq i \} \) \( \setminus \{ j : \hat{r}_j = \hat{r}_l, j \neq i \} = l \). This implies

\[ \sum_{j: x_j^* = \hat{r}_l} b_j - \sum_{j: x_j^* = \hat{r}_l} b_j = -b_l. \]

Also, \( \{ j : r_j^* \neq \hat{r}_l \} \setminus \{ j : \hat{r}_j \neq \hat{r}_l \} = l \). Thus, we can write

\[ \sum_{j: x_j^* \neq \hat{r}_l} c_{ij} - \sum_{j: x_j^* \neq \hat{r}_l} c_{ij} = c_{il}, \]
which implies,

\[
\sum_{i : i \neq l} C_i(r^*) - \sum_{i : i \neq l} C_i(\tilde{r}) = \sum_{i : i \neq l} \left( -b_i \left( \frac{b_i}{w_{r_i^*}} \right) + \frac{\mu}{2} c_{il} \right).
\]

Using a similar approach, we can show:

\[
\sum_{i : i \neq l} C_i(r^*) - \sum_{i : i \neq l} C_i(\tilde{r}) = \sum_{i : i \neq l} \left( b_i \left( \frac{b_i}{w_{r_i^*}} \right) - \frac{\mu}{2} c_{il} \right).
\]

So,

\[
\left( \sum_{i : i \neq l} C_i(r^*) - \sum_{i : i \neq l} C_i(\tilde{r}) \right) + \left( \sum_{i : i \neq l} C_i(r^*) - \sum_{i : i \neq l} C_i(\tilde{r}) \right) =
\]

\[
\sum_{i : i \neq l} \left( -b_i \left( \frac{b_i}{w_{r_i^*}} \right) + \frac{\mu}{2} c_{il} \right) + \sum_{i : i \neq l} \left( b_i \left( \frac{b_i}{w_{r_i^*}} \right) - \frac{\mu}{2} c_{il} \right).
\]

Now consider

\[
C_l(r^*) - C_l(\tilde{r}) = b_l \sum_{i : r_i^* = r_l^* \text{ and } i \neq l} b_i \left( \frac{b_i}{w_{r_i^*}} \right) + \frac{\mu}{2} \sum_{i : i \neq l} c_{il}
\]

\[
-b_l \sum_{i : r_i = r_l} b_i \left( \frac{b_i}{w_{r_i^*}} \right) - \frac{\mu}{2} \sum_{i : i \neq l} c_{il}.
\]

We know that,

\[
\sum_{i : r_i^* \neq r_l^* \text{ and } i \neq l} c_{il} - \sum_{i : r_i \neq r_l} c_{il} = \sum_{i : r_i^* \neq r_l^* \text{ and } i \neq l} c_{il} + \sum_{i : r_i^* \neq r_l^* \text{ and } i \neq l} c_{il} - \sum_{i : r_i^* \neq r_l^* \text{ and } i \neq l} c_{il}.
\]
However,

\[ \sum_{i: r_i^* \neq r_i' \atop r_i' \neq r_i} c_{il} = \sum_{i: r_i^* \neq r_i' \atop r_i' \neq r_i} c_{il}. \]

Thus we see:

\[ \sum_{i: r_i^* \neq r_i' \atop r_i' \neq r_i} c_{il} - \sum_{i: r_i^* = r_i' \atop r_i' \neq r_i} c_{il} = \sum_{i: r_i^* = r_i' \atop r_i' \neq r_i} c_{il} - \sum_{i: r_i^* = r_i' \atop r_i' \neq r_i} c_{il} \]

\[ \Rightarrow C_l(r^*) - C_l(\hat{r}) = -b_l \sum_{i: r_i^* = r_i' \atop r_i' \neq r_i} \frac{b_i}{w_{\hat{r}_i}} + \frac{\mu}{2} \sum_{i: r_i^* = r_i' \atop r_i' \neq r_i} c_{il} + b_l \sum_{i: r_i^* = r_i' \atop r_i' \neq r_i} \frac{b_i}{w_{r_i^*}} - \frac{\mu}{2} \sum_{i: r_i^* = r_i' \atop r_i' \neq r_i} c_{il}. \]

But we also know that \( \{i: \hat{r}_i = \hat{r}, i \neq l\} = \{i: r_i^* = r_i, i \neq l\} \) and \( \{\hat{r}_i = r_i^*, i \neq l\} = \{r_i^* = r_i^*, i \neq l\} \). So,

\[ C_l(r^*) - C_l(\hat{r}) = -b_l \sum_{i: r_i^* = r_i' \atop r_i' \neq r_i} \frac{b_i}{w_{\hat{r}_i}} + \frac{\mu}{2} \sum_{i: r_i^* = r_i' \atop r_i' \neq r_i} c_{il} + b_l \sum_{i: r_i^* = r_i' \atop r_i' \neq r_i} \frac{b_i}{w_{r_i^*}} - \frac{\mu}{2} \sum_{i: r_i^* = r_i' \atop r_i' \neq r_i} c_{il} \]

\[ = \sum_{i: i \neq l \atop r_i^* = \hat{r}_i} \left( -b_l \left( \frac{b_i}{w_{\hat{r}_i}} \right) + \frac{\mu}{2} c_{il} \right) + \sum_{i: i \neq l \atop r_i^* = r_i^*} \left( b_l \left( \frac{b_i}{w_{r_i^*}} \right) - \frac{\mu}{2} c_{il} \right) \]

\[ = \left( \sum_{i: i \neq l \atop r_i^* = \hat{r}_i} C_l(r^*) - \sum_{i: i \neq l \atop \hat{r}_i = \hat{r}_i} C_l(\hat{r}) \right) + \left( \sum_{i: i \neq l \atop r_i^* = r_i^*} C_l(r^*) - \sum_{i: i \neq l \atop \hat{r}_i = r_i^*} C_l(\hat{r}) \right). \quad (3) \]

The last step follows from (1). Thus we have shown:

\[ \left( \sum_{i: i \neq l \atop r_i^* = \hat{r}_i} C_l(r^*) - \sum_{i: i \neq l \atop \hat{r}_i = \hat{r}_i} C_l(\hat{r}) \right) + \left( \sum_{i: i \neq l \atop r_i^* = r_i^*} C_l(r^*) - \sum_{i: i \neq l \atop \hat{r}_i = r_i^*} C_l(\hat{r}) \right) \quad (4) \]

\[ = C_l(r^*) - C_l(\hat{r}) < 0 \]

This completes the proof for STEP 3.

Hence using Steps 1, 2 and 3, \( C_0(r^*) - C_0(\hat{r}) = 2(C_l(r^*) - C_l(\hat{r})) < 0 \). This contradicts our assumption that \( \hat{r} \) is a globally optimal solution of \( C_0 \). Hence \( \hat{r} \) is
also a Nash equilibrium in pure strategies.


Vita
Aditya Kurve

Education
2009.08 to 2011.08 MS in Electrical Engineering, The Pennsylvania State University
2002.09 to 2006.07 BE in Electronics, University of Pune, India

Selected Journal Publications


- Aditya Kurve, David J. Miller and George Kesidis, Multicategory Crowdsourcing Accounting for Plurality in Worker Skill and Intention, Task Difficulty, and Task Heterogeneity, IEEE Transactions on Knowledge and Data Engineering (Under Revision), June 2013.