ALGORITHMIC, GAME THEORETIC AND LEARNING THEORETIC ASPECTS OF DISTRIBUTED OPTIMIZATION

A Thesis
Presented to
The Academic Faculty

by

Steven Ehrlich

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Algorithms Combinatorics and Optimization in the
College of Computing

Georgia Institute of Technology
January 15, 2016

ALGORITHMIC, GAME THEORETIC AND LEARNING THEORETIC ASPECTS OF DISTRIBUTED OPTIMIZATION

Dedidcated to Nathan Ehrlich, sorry it's a little late.

ACKNOWLEDGEMENTS

I could not have completed this thesis without a lot of help from a lot of people. In particular, I'd like to thank Albert Bush, Robert Krone, Peter and Krista Whalen for being the best part of living in Atlanta. To the myriad people in California who gave me excuses to come visit, it was and is much appreciated. I was glad to have Chris Berlind, Florin Constantin, and Yingyu Liang as coauthors. My parents and siblings were sympathetic when I needed it and unsympathetic when I needed that instead. Lastly, special thanks are due my advisors Nina Balcan and Jeff Shamma from whom I learned much.

```
The road to wisdom?
  Well it's plain
  And simple to express.
Err,
  And err,
  And err again ---
But less,
  And less,
  And less.
```

- Piet Hein

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Distributed systems are fundamental to today's world. Many modern problems involve multiple agents either competing or coordinating across a network, and even tasks that are not inherently distributed are often divided to accommodate today's computing resources. In this thesis we consider distributed optimization through the lens of several problems.

We first consider the fragility of distributed systems, with an investigation in game theory. The inefficiency, relative to total cooperation, of agents acting myopically in their own interest is well studied as the so called the Price of Anarchy. We assess how much further the social welfare can degrade due to repeated small disruptions. We consider two models of disruptions. In the first, agents perceive costs subject to a small adversarial perturbation; in the second a small number of Byzantine players attempt to influence the system. For both models we improve upper and lower bounds on how much social welfare can degrade for several interesting classes of games.

We next consider several problems in which agents have partial information and wish to efficiently coordinate on a solution. We measure the cost of their coordination by the amount of communication the agents must exchange. We next investigate a problem in active and semi-supervised learning. After providing a novel algorithm to learn it in the centralized case, we consider the communication cost of this algorithm when the examples are distributed amongst several agents. We then turn to the problem of clustering when the data set has been distributed among many agents. Here we devise an algorithm for coordinating on a global approximation that can be communicated efficiently by the use of coresets. Finally we consider a problem of submodular maximization where the objective function has been distributed among agents. We adapt a centralised approximation algorithm to the distributed setting with efficient communication between the agents.

CHAPTER I

INTRODUCTION

As computing is becoming ubiquitous we face new challenges in the realm of distributed computation. There are many reasons a given problem might be distributed. Sometimes the problem is inherently distributed — in sensor networks, data collection happens simultaneously by disparate sensors and the data subsequently must be combined. Other times a problem is broken up to meet technical constraints. Because computing is so cheap today we can tackle problems in ways we could not before, with clusters replacing the supercomputers of yesterday.

It is important to solve our problems efficiently. Of course any attempt to optimize can only happen with respect to some measure. Classically the default cost considered is temporal, i.e. we measure an algorithm that computes something by the maximum amount of time it could take to complete the computation. Another foundational model measures the spatial costs of computation — how much memory is consumed. In the case of approximation algorithms, when we cannot guarantee finding a solution at all, we instead measure how accurate the solution is. For machine learning problems, we often measure performance by the number of examples that are needed to (guarantee that we) reach a given accuracy threshold.

In many distributed problems, the natural cost we wish to optimize for is communication. For any particular distributed problem we want to find solutions while enabling the parties to communicate as efficiently as possible. Indeed the cost of coordination for parties with asymmetric information is what distinguishes a problem in the distributed setting from a centralized one.

One prominent setting is Game Theory which features problems that are inherently distributed. The standard game theoretic setting contains many individual agents interacting in a common environment, each trying to optimize their own objective function which depends on their joint action. In some sense, this is an optimization problem where the objective is distributed into multiple (possibly conflicting) parts. These are then passed out to distinct agents who each attempt to optimize their portion. If we consider a particular measure of how good the group outcome is — a so-called measure of *social welfare* — we can compare the distributed problem in which the agents act independantly to a centralized one. The standard setting compares the social welfare in the optimum state to the social welfare in a Nash equilibrium. The ratio of these provides a measure of the ineffiency introduced by decentralization. We consider a variation on this classic work where all the agents are subject to uncertainty in their environment. We seek to measure the additional inefficiency caused by this uncertainty. We consider two types of noisy environments: one in which the agents have imprecise measurements of their utility functions, and one in which there are a few agents with irregular or malicious behavior.

We also consider several instances in which a learning or optimization problem can be solved or approximated in the standard centralized setting, but the problem is distributed in some way. We then develop tools and algorithms that allow for efficient coordination. In particular we consider the problems of clustering and (for both the semi-supervised and active settings) of learning two-sided disjunctions. We shall describe the problems in more detail below.

Noise in Distributed Settings: Game Theory is the study of multiple agents interacting in a shared envionment. It is common to assume that agents are self-interested and seek to maximize their own utility functions. The concept of a Nash equilibrium refers to a state in which each agent would decrease their utility if they *unilaterally* changed their strategy. However, such an equilibrium might be suboptimal for each of the parties involved. Using the sum of all players utilities as a measure of social welfare, we can compare the social cost of an equilbrium with that of the optimal outcome. The most famous such measure is the Price of Anarchy (PoA)[59] which is the ratio of the social welfare in the optimal state to the social welfare of the 'worst' Nash equilibrium.

In real world settings the model's assumptions may prove inaccurate. Chief among these assumptions are that, when the agents decide their action, they know what payoffs they recieve given what other agents select. This may not be correct for a number of reasons. An agents utility may depend on small order terms ignored by the model. Consider a job-scheduling game in whch we measure how long it takes a particular set of programs to run. Even though the cost will be dominated by how the jobs are distributed among the machines, other factors will also contribute to the runtime, e.g. the operating system will have a small effect which is ignored in the model's cost assumption. Similarly, consider a congestion game in a road network. For any given car, the time to travel along a particular path may be primarily due to congestion, but there will many small factors which contribute to the precise time — even things like traffic light timings or the weather. In some games, the utility measurement may even be the result of a random or statistical process. Thus by seeking to measure the effects of smaller order terms we are measuring the fragility of our model — does it still hold up under more realistic conditions? In particular, we are interested in how the dynamics change when agents respond in noisy conditions, when they might misjudge, if only slightly, their outcomes. This challenge led to a measure called the Price of Uncertainty (PoU), first analyzed by Balcan Blum and Mansour.[12]

When agents who begin in some state and follow a set of dynamics (best response) to what they percieve to be their outcomes at every time step, the social welfare evolves over time. The PoU measures the largest possible increase, and provides a measure of how resilient the game and dynamics are to small perturbations. [1]

Another implicit assumption is that each player is utility maximizing. Imagine a congestion game with drivers. Most drivers try to minimize their time in the car, but others may be on the roads for other reasons — they want to take a scenic route, they just enjoy driving. Recently, taxi drivers have taken to clogging traffic as a protest against competition. Social networks can suffer from Sybil attacks, where attackers create and manage multiple profiles with the goal of amplifying their effect on the network. Agents of this sort are called Byzantine after the Byzantine Generals Problem. To analyze this we define the Price of Byzantine Behavior (PoB). For a particular class of games following a particular class of

[1]We slightly alter the definition from reference so that $PoU(0)$ is a nontrivial question. As in the original work, we focus on games where $PoU(0)$ is small, i.e. a logarithmic increase.

dynamics, the PoB is the maximum ratio between the cost of an initial starting state and a state that can be reached when there are at most b Byzantine players.

We consider potential games following improved response dynamics. Furthermore we focus on those where there is a small GAP between potential and social welfare. These conditions taken together tell us the dynamics we are studying are not inherently fragile — thus any fragility in these dynamics can be attributed to the presence of noise. Our results provide novel upper and lower bounds in the PoU setting and are tight in the PoB setting.

Learning Two-Sided Disjunctions with Unlabelled Examples: We next consider a problem in learning theory. The PAC model (Probably, Approximately Correct)[85] of Valiant allows for guarantees on the sample complexity. Following real world trends, theoretical models incorporating unlabelled data have been developed. In particular Balcan and Blum[7] established a theoretical framework for semisupervised learning. Assumptions about the relationship between the data distribution and the concept class were formalized as compatibility notions. This allows a measurement of how well unlabelled data fits with a particular hypothesis. Semisupervised learning gives the learner access to two pools of examples, one labelled, another larger and unlabelled. We measure the number of examples of each type separately. Active learning allows the learner to select which examples they wish to have labelled, and measures the number of queries to the labelling oracle.

In these settings, similar to the PAC model, generalization is well-understood; given sufficient data, hypotheses which are both consistent with labelled examples and compatible with the unlabelled ones are guaranteed to generalize well. However finding hypotheses both consistent and compatible with a particular data set is much harder. We consder the problem of learning two-sided disjunctions: all positive examples contain a feature that is a positive indicator and all negative examples include a feature which is a negative indicator. In the fully supervised setting this problem is equivalent to the simple problem of learning disjunctions. However, in the semi-supervised setting it was an open problem to reduce the number of labelled examples needed. We provide a novel algorithm in both the semi-supervised and active settings for finding consistent, compatible hypotheses. We conclude this section with a discussion of how to turn the centralized algorithm into a distributed one.

In distributed learning settings, access to the underlying distribution (in the form of examples) is distributed amongst multiple agents. This naturally introduces a need to communicate efficiently.

Minimizing Communication in Optimization Problems: We next turn to the problem of clustering. Given a set of points and some similarity metric, the goal of the clustering problem is to divide the points into some fixed number of clusters such that the points in the clusters are 'close' to each other given the similarity metric. More formally, the goal is usually to minimize the size of the largest cluster given some notion of size. There are many choices for how to measure cluster size; two of the most prominent are k-means and k-medians. We note in passing that these problems are hard — given a set of points it is NP hard to find the optimal division into k clusters such that the mean of the largest cluster is minimized.[2] In practice there are a number of heuristic algorithms that work well, e.g. Lloyd's algorithm, which iteratively relabels all points as belonging to the cluster with the closest center.

We examine this problem in the distributed setting. Consider agents who independently have been collecting parts of a larger data set and wish to work together to find an optimal clustering. Of course the simplest thing to do in this problem is for each agent to share all their data, reducing this to the centralized setting. However this may not be possible for a number of reasons (e.g. sharing data is costly, or privacy concerns.) We want a solution to this problem as good as can be found in the centralized setting (or at least an approximation thereof). Furthermore, we wish to do so as parsimoniously as possible. The key idea in the algorithm we develop is the use of coresets[39]. Intuitively, each agent finds a way to compress their data into a representative sample with a much smaller size. A coreset of a given set of points will be a weighted point set such that, for any set of centers, the cost of the original data set is well approximated by the cost of the coreset. Using these we

[2] The underlying assumption is that the target clustering (the ground truth) is close to the k-means or k-medians minimum. Balcan, Blum and Gupta showed that if all clusterings that are close to the target (in the sense of the labelling of the points) are also close according to these metrics, then one can sidestep finding the best k-means solution and instead find nearly optimal clusterings directly.

construct a data set which, though considerably smaller, well-approximates the original global data set.

We finish with a discussion of ongoing work related to distributed submodular maximization. A set-valued function $f : 2^{\{n\}} \to \mathbf{R}$ is said to be submodular if it has diminishing marginal returns. Formally, for $A \subseteq B$ and $x \notin B$, $f(A + x) - f(A) \geq f(B + x) - f(B)$. Many important problems can be represented by submodular functions, such as cuts in graphs or cost of spanning trees. We focus on submodular maximization problems with cardinality constraints. It is well known that this can be approximated with a simple greedy algorithm.

Our goal here is an algorithm that allows for optimization of the global objective with efficient communication. To do this we aim to decentralize the algorithm by replacing submodular function oracle queries with approximations taken by sampling the agents partial functions. We present some preliminary results on this topic which allow for distributed optimization with communication only logarithmically dependent on the number of nodes.

Summary of Contributions: The remainder of the thesis is organized as follows:

Chapter 2: We provide improved upper and lower bounds for the Price of Uncertainty and Price of Byzantine Behavior for several classes of potential games. Most notably, we provide a tight bound on the PoB for consensus games, and an exponential improvement over the previous best results for the PoU of congestion games. This work was published in WINE, and is joint work with Nina Balcan and Florin Constantin.[13]

Chapter 3: We provide novel algorithms capable of finding consistent, compatible hypotheses for the class of two-sided disjunctions, allowing us to efficiently make use of unlabelled examples in both the semi-supervised and active settings. We extend these to the distributed setting and analyze the communication costs incurred. This work was published in ICML, and is joint work with Nina Balcan, Chris Berlind, and Yingyu Liang.[8]

Chapter 4: We provide an algorithm for clustering a distributed data set with the aim of minimizing communication cost. Our primary contribution is adopting the notion of coresets (which have previously been used for clustering) to the distributed setting. This work was published in NIPS, and is joint work with Nina Balcan and Yingyu Liang.[14]

Chapter 5: We provide a distributed algorithm which maximizes a monotone submodular optimization problem subject to cardinalty constraints. We achieve an optimal approximation with communication cost that is independent of the number of number of nodes and cardinalty constraint.

CHAPTER II

PRICE OF UNCERTAINTY AND PRICE OF BYZANTINE BEHAVIOR

Uncertainty is present in different guises in many settings, in particular in environments with strategic interactions. However, most game-theoretic models assume that players can accurately observe interactions and their own costs. In this chapter we quantify the effect on social costs of two different types of uncertainty: adversarial perturbations of small magnitude to costs (effect called the Price of Uncertainty [12]) and the presence of several players with Byzantine, i.e. arbitrary, behavior (effect called the Price of Byzantine behavior). We provide lower and upper bounds on both the Price of Byzantine behaviour and the Price of Uncertainty in well-studied classes of potential games including consensus games, set-covering games, and (λ, μ)-smooth games. Many of our bounds are tight and significantly improve over the previously best known bounds of [12].

2.1 Introduction

Overview Uncertainty, in many manifestations and to different degrees, arises naturally in applications modeled by games. In such settings, players can rarely observe accurately and assign a precise cost or value to a given action in a specific state. For example a player who shares costs for a service (e.g. usage of a supercomputer or of a lab facility) with others may not know the exact cost of this service. Furthermore, this cost may fluctuate over time due to unplanned expenses or auxiliary periodic costs associated with the service. In a social network, the cost a player perceives for an action such as disagreeing with some neighbor may also fluctuate over time. Another type of uncertainty arises when some players are misbehaving, i.e., they are Byzantine, perhaps due to incentives that have not been accurately modeled in the game.

In this chapter we assess the long-term effect of small local uncertainty on natural dynamics in cost-minimization *potential games* [66]. We study the degree to which small fluctuations in costs or the presence of Byzantine players can impact the result of natural best-response and improved-response dynamics in well-studied classes potential games, using the framework introduced by [12]. A first class we analyze is that of *consensus games* [24, 26, 66, 80, 82]. We show that in these games uncertainty can have a strong *snowball effect*, analogous to the increase in size and destructive force of a snowball rolling down a snowy slope. Namely, we show that small perturbations of costs on a per-player basis or a handful of players with Byzantine (i.e. adversarial) behavior can cause a population of players to go from a good state (even a good equilibrium state) to a state of much higher cost. A second class we analyze is that of *set-covering games* [24]. We improve on the previously known bounds of Balcan, Blum, and Mansour [12] for these games and show that they are more resilient to uncertainty. Finally, we also provide positive bounds for (λ, μ)-smooth games using random order. Thus our work provides a clear picture of the consequences of uncertainty or unmodeled low-order effects on the behavior of players in these important classes of games.

Problem setup and results We consider both *improved-response* (IR) dynamics in which at each time step exactly one player may update strategy in order to lower his (apparent) cost and *best-response* (BR) dynamics in which the updating player chooses what appears to be the least costly strategy. Any state is assigned a *social cost*, which in this chapter is defined as the sum of all players' costs in that state. We measure the effect of uncertainty as the maximum multiplicative increase in social cost when following these dynamics. We instantiate this measure to each type of uncertainty.

For the first uncertainty type, we assume adversarial perturbations of costs of magnitude at most $1 + \epsilon$ for $\epsilon > 0$ (a small quantity that may depend on game parameters). That is, a true cost of c may be perceived as any value within $[\frac{1}{1+\epsilon}c, (1 + \epsilon)c]$.[1] Consider a game G and an initial state S_0 in G. We call a state S (ϵ, IR)-*reachable* from S_0 if there exists a valid ordering of updates in IR dynamics and corresponding perturbations (of magnitude at most ϵ) leading from S_0 to S. The *Price of Uncertainty* [12] (for IR dynamics) given ϵ of game G is defined as the ratio of the highest social cost of an (ϵ, IR)-reachable state S to

[1]As mentioned in [12] this is similar to the Statistical Query model of Kearns. [55]

the social cost of starting state S_0.

$$PoU_{IR}(\epsilon, G) = \max\left\{\frac{cost(S)}{cost(S_0)} \;:\; S_0;\, S\ (\epsilon, IR)\text{-reachable from } S_0\right\}$$

For a class $\mathcal{G}$ of games and $\epsilon \geq 0$ we define $PoU_{IR}(\epsilon, \mathcal{G}) = \sup_{G\in\mathcal{G}} PoU_{IR}(\epsilon, G)$ as the highest PoU of any game G in $\mathcal{G}$ for ϵ. PoU_{BR} is defined analogously.

For the second uncertainty type, we assume B additional[2] players with arbitrary, or Byzantine [69] behavior. Similar to the Price of Uncertainty, the *Price of Byzantine behavior* ($PoB(B)$) on social cost measures the effect of B Byzantine players on social cost, and is defined as the maximum ratio of the cost of a state reachable in the presence of B Byzantine agents to that of the starting state.

$$PoB(B, G) = \max\left\{\frac{cost(S)}{cost(S_0)} \;:\; S_0;\, S\ B\text{-Byz-reachable from } S_0\right\}$$

where state S of G is *B-Byz-reachable* from S_0 if some valid ordering of updates by players (including the B Byzantine ones) goes from S_0 to $S. PoB(B, \mathcal{G}) = \sup_{G\in\mathcal{G}} PoB(B, G)$ for class $\mathcal{G}$. PoB, like PoU, may depend on the dynamics.[3]

Note that for any class of games $\mathcal{G}$, we know that $PoU(0, \mathcal{G}) = PoB(0, \mathcal{G})$.

In the games we study, social costs cannot increase much without uncertainty (namely $PoU(0)$ and $PoB(0)$ are small). Thus if the PoU or PoB is low, then one can have confidence that small errors by players in estimating costs or arbitrary behavior of others will not cause a system currently in a low-cost state to devolve into one of much higher cost. On the other hand, a high PoU or a high PoB reveals a high sensitivity to such factors and raises a warning flag for designers or modelers of these systems.

Next we introduce the classes of games we study and summarize our results.

Consensus games [29] model a basic strategic interaction: choosing one side or the other (e.g. in a debate) and incurring a positive cost for interacting with each agent that chose the other side. More formally, there are two colors (or strategies), white (W) and red (R), which each player may choose; hence IR and BR dynamics coincide. Each player occupies a

[2]Note that the presence of additional players alters the game. It is necessary that the altered game with these additional players remains in the class $\mathcal{G}$.

[3]We omit parameters from PoU and PoB when they are clear from context.

different vertex in an undirected graph G with vertices $\{1,\ldots,n\}$ and edges $E(G)$ (without self-loops). A player's cost is defined as the number of neighbors with the other color.

Bounds on $PoU(\epsilon)$ We establish $PoU(\epsilon) = \Omega(n^2\epsilon^3)$ for $\epsilon = \Omega(n^{-1/3})$ and $PoU(\epsilon) = O(n^2\epsilon)$ for any ϵ. We note that these bounds are asymptotically tight as $\Theta(n^2)$ for constant ϵ, and the highest possible since costs are in $[1, n^2]$.

Bounds on $PoB(B)$ We provide a tight bound of $PoB(B)$ as $\Theta(n\sqrt{nB})$ for B Byzantine players; in contrast, individual updates by cost-minimizing players cannot increase the social cost. To achieve this, we identify a stylized game with the essential properties of a consensus game with B Byzantine players. While the cost in a consensus game depends on the state, in our new game it suffices to bound the number of edges. This is achieved by demonstrating an explicit extremal graph with $\Theta(n\sqrt{nB})$ edges and showing via an inductive argument that an arbitrary game in our new class can be reduced (without losing any edges) to our extremal graph. We note that our new bound greatly improves [12]'s bound of $\Omega(n)$ for $B = 1$.

Set-covering games [24] model many applications where all users of a resource share fairly its base cost. These natural games fall in the widely studied class of fair-cost sharing games [2]. In a set-covering game, there are m sets, each set j with its own fixed weight (i.e. base cost) w_j. Each of the n players must choose exactly one set j (different players may have access to different sets) and share its weight equally with its other users, i.e incur cost $w_j/n_j(S)$ where $n_j(S)$ denotes the number of users of set j in state S. Prior work has provide a tight characterization of $PoB_{BR}(1) = \Theta(n)$ for such games so we focus on providing bounds on $PoU_{IR}(\epsilon)$ and $PoU_{BR}(\epsilon)$.

Bounds on $PoU_{IR}(\epsilon)$ We prove $PoU_{IR}(\epsilon) = (1+\epsilon)^{O(m^2)}O(\log m)$ for $\epsilon = O(\frac{1}{m})$. This is the first bound known that has no dependence on the number of players n, and so will be small in the natural case that the number of players is large and the number of resources is small (or even constant).[4] We also improve the existing lower bound for

[4]Our approach yields a new upper bound on PoU_{BR} in a class of *matroid congestion games* (see Section 2.5.1.1 for more details).

these games (due to [12]) to $PoU_{IR}(\epsilon) = \Omega(\log^p m)$ for $\epsilon = \Theta(\frac{1}{m})$ and any constant $p > 0$. Our new lower bound is a subtle construction that uses an intricate "pump" gadget with finely tuned parameters. A pump replaces, in a non-trivial recursive manner with identical initial and final pump states, one player in a state of small cost with one player in a state of high cost. This gadget can be adapted for a polylogarithmic lower bound on PoU_{IR} (again for $\epsilon = \Theta(\frac{1}{m})$) in a much broader class of games.

Bounds on $PoU_{BR}(\epsilon)$ We show a lower bound of $PoU_{BR}(\epsilon) = \Omega(\epsilon n^{1/3}/\log n)$ for $\epsilon = \Omega(n^{-1/3})$ and $m = \Omega(n)$ which is valid for any ordering of updates in which each player moves sufficiently often, unlike the bound of [12] which only showed the existence of some bad ordering.

(λ, μ)-smooth games. Finally, we upper bound the effect of arbitrary perturbations assuming a random order of player updates in an important class of games, (λ, μ)-*smooth games* [82]. (λ, μ)-smooth games are those in which players unilateral cost deviations from a state S to S' in total do not cost significantly more than the costs of states S and S'. We show that these games are resilient to uncertainty provided that player who best responds at each time step is chosen randomly.

We note that our lower bounds use judiciously tuned gadgets that create the desired snowball effect of uncertainty. Most of them hold even if players must update in a specified order, e.g. round-robin (i.e. cyclically) or the player to update is the one with the largest reduction in (perceived) cost. Our upper bounds on PoU hold no matter which player updates at any given step.

2.2 Related Work

2.2.1 Potential Games

Potential games are an important class of games that have been studied extensively [38, 72, 73, 87]. They were first introduced in 1973 by Rosenthal [80] as games which have a *potential function.* An (exact) potential function is a function on joint actions such that an agent's change in utility when responding to some joint action will be equal to difference in the potential function. He proved these games had Pure Nash Equilibria, by noting that any local minimum of the potential function would be a Pure Nash Equilibrium. Monderer and Shapley proved that exact potential games correspond precisely to congestion games [66].

Many of the games considered in this chapter are matroid congestion games, in which each player's strategy must be a base of a matroid. Ackermann et al. [1] show that all best-response sequences in such games (with perfect, accurate information) are of polynomial length. They also show that many non-matroid congestion games can have best-response paths of exponential length.

Anshelevich et al. [2] were the first to apply the potential function method to fair-cost sharing games, a generalization of set-covering games in which any player must connect his source and destination in a graph with a path. A player may use more than one edge and shares the cost of each such edge fairly with all other users of the edge. These games are have been used in a wide variety of contexts [76, 83, 37]. Another important class of potential games are consensus games which have also been extensively studied [11, 29, 68, 70].

2.2.2 Uncertainty and Related Solution Concepts

Uncertainty in potential games. There are relatively few papers that address the effects of uncertainty in potential games. Balcan, Blum, and Mansour [12] defined the Price of Uncertainty, recognizing the harmful effect on social costs of imperfect information coupled with selfish behavior. They showed lower and upper bounds on PoU for BR or IR dynamics, adversarial or random order of strategy updates and for several classes of matroid congestion games. Unlike us, they very briefly touch on Byzantine players or consensus games. Monderer et al. [67] study potential games in which players face uncertainty about the number of other players. They define and prove existence and uniqueness of a certain equilibrium concept in any such game, and they show that this equilibrium can be better if players have less information. Similarly, Meir, Tennholtz, Bachrach and Key [64] study congestion games in which some players may fail to participate in the game. In this model they show that independent failure of the agents improves games by avoiding bad equilibria.

Roth [81] shows tight bounds for the social cost of players using regret-minimization dynamics in the presence of Byzantine players in congestion games where resources have linear costs. This work measures how much the quality of the worst coarse-correlated equilibria can degrade with the introduction of Byzantine players. In contrast, we measure how much the dynamics can degrade the social cost of any given state.

Similar Solution Concepts. Some work has been done to investigate how much the social cost can increase following a particular type of dynamics, i.e. for an uncertainty level of 0. The Price of Sinking, introduced by [42] tries to investigate best response dynamics in games which don't necessarily have a pure Nash equilibrium, by finding instead sets of states in which best response dynamics get trapped (the so called sinks) and analyzing the expected social welfare over these states. Recent work in [17] similarly explores different best-response dynamics, and defines the dynamic inefficiency as the ratio of the social optimum to the average social welfare over time. Note that dynamic inefficiency does not consider uncertainty and it is rather an average-case measure as opposed to PoU or PoB, that are worst-case.

Our technique to bound the PoB in consensus games produces a setting similar to that of diffusion models. We introduce the notion of B-flippablity as a consensus game in which flipping B agents can change every player from red to blue. The study of diffusion models [23, 57] investigates the spread of a change through a network given that a node will adopt the change once some threshold of neighbors have already accepted it. Blume, Easley, Kleinberg, Kleinberg and Tardos [23] investigate the diffusion through regular graphs given that the failure thresholds for each node are drawn i.i.d.; the graphs they consider need not be finite. Kempe, Kleinberg and Tardos [56, 57] consider the question of which nodes in the graph have the maximum influence over the rest. We believe our constructions can provide new insights into diffusion models as well.

2.3 The Formal Model

We consider games with n cost-minimizing selfish players. A player i can use a finite set Σ_i of strategies and independently aims to minimize his *cost*. The dependence of his cost on his and others' strategies is given by $cost_i : \times_{i'=1}^{n} \Sigma_{i'} \to \mathbb{R}$. A *state* is a strategy vector $S = (S_1, \ldots, S_n)$ where $S_i \in \Sigma_i$ is a strategy of player i for all $i \in [n] = \{1, \ldots, n\}$. For a joint action S we denote by S_{-i} the actions of players $j \neq i$, i.e. $S_{-i} =$

$(S_1, S_2, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n)$. The social cost $cost(S)$ in a state S aggregates individual costs $cost_1(S), \ldots, cost_n(S)$. From a global point of view, lower social costs are better. We consider as a social cost function the sum of the costs of all players: $cost(S) = \sum_{i=1}^{n} cost_i(S)$, which is a common social cost in the literature.

Dynamics and potential games. We define natural dynamics for any player to lower his current cost. We say that player i performs *improved-response* (IR; equivalently, better-response), if i updates its strategy S_i to S'_i, an improved strategy (i.e. $cost_i(S_i, S_{-i}) \geq cost_i(S'_i, S_{-i})$), given S_{-i}. We say that player i performs *best-response* (BR) if i updates its strategy S_i to S'_i, the best strategy (i.e. $cost_i(S'_i, S_{-i}) \leq cost_i(S''_i, S_{-i}), \forall S''_i \in \Sigma_i$) given S_{-i}. BR is a type of IR dynamics. A state $(S_1, \ldots, S_n)$ is a *Nash equilibrium* if each player minimizes his cost given others' strategies, i.e. if $cost_i(S_i, S_{-i}) \leq cost_i(S''_i, S_{-i})$ for any player i and strategy $S''_i \in \Sigma_i$. That is, a Nash equilibrium S is a stable point for either dynamics: S_i is a best-response to $S_{-i}, \forall i$.

The games we study in this chapter are exact potential games. We say that game G has an (exact) *potential function* $\Phi : \times_{i'=1}^{n} \Sigma_{i'} \rightarrow \mathbb{R}$ if the change in a player's cost at any strategy update (not necessarily cost-reducing) equals the change in potential, i.e. if

$$cost_i(S_i, S_{-i}) - cost_i(S'_i, S_{-i}) = \Phi(S_i, S_{-i}) - \Phi(S'_i, S_{-i}),\ \forall i \in [n], \forall S_i, S'_i \in \Sigma_i, \forall S_{-i} \in \Sigma_{-i}.$$

The potential Φ thus globally tracks individual strategy updates. As the potential decreases at any update in IR (and BR) dynamics in the absence of uncertainty, these dynamics are guaranteed to converge. Such games have been widely studied in the literature. [2, 24, 26, 28, 66, 80, 82].

PoU. We now introduce a central concept in this chapter: the *Price of Uncertainty* (PoU) [12] for a class of games $\mathcal{G}$, measuring the degradation of social costs during dynamics based on perturbed costs. Fix $\epsilon > 0$, that may depend on parameters common to all games in $\mathcal{G}$, e.g. $\epsilon = 1/n$. We consider Nash dynamics in which players observe costs that are perturbed, possibly adversarially, by a multiplicative factor in $[\frac{1}{1+\epsilon}, 1 + \epsilon]$. That is, a player whose actual cost in state S is c may perceive instead any cost in the interval $[\frac{1}{1+\epsilon}c, (1 + \epsilon)c]$. Consider a game $G \in \mathcal{G}$ and an initial state S_0 in G. We call a state S (ϵ, IR)-*reachable* from

S_0 if there exists a valid ordering of updates in IR dynamics and corresponding perturbations (of magnitude at most ϵ) leading from S_0 to S. We are now ready to define the price of uncertainty for IR dynamics given ϵ of game G as the ratio of the highest social cost of an (ϵ, IR)-reachable state S to the social cost of starting state S_0.

$$PoU_{IR}(\epsilon, G) = \max\left\{\frac{cost(S)}{cost(S_0)} : S_0; S\ (\epsilon, IR)\text{-reachable from } S_0\right\} \text{ and}$$
$$PoU_{IR}(\epsilon, \mathcal{G}) = \sup_{G\in\mathcal{G}} PoU_{IR}(\epsilon, G)$$

The PoU of class $\mathcal{G}$ is defined as the highest PoU of any game G in $\mathcal{G}$ for the same ϵ. For the more specific best-response dynamics, PoU_{BR} is defined analogously.

PoB. Instead of costs, players are often uncertain about other players' goals. A worst-case assumption is that some players can behave in a *Byzantine* [21, 69], i.e. arbitrary, manner[5]. A natural measure of the effect of B Byzantine players from a starting state S in game G is the ratio of the cost of a state reachable in the presence of B Byzantine agents to that of S. The maximum of the ratios over all states, that we call the *Price of Byzantine behavior*, abbreviated PoB, is clearly analogous to the Price of Uncertainty.

$$PoB(B, G) = \max\left\{\frac{cost(S)}{cost(S_0)} : S_0; S\ B\text{-Byz-reachable from } S_0\right\} \text{ and}$$
$$PoB(B, \mathcal{G}) = \sup_{G\in\mathcal{G}} PoB(B, G)$$

where state S of G is *B-Byz-reachable* from state S_0 of G if some valid order of updates by players (including the B Byzantine ones) goes from S_0 to S. PoB, like PoU, may depend on the dynamics used[6]. We note that the B Byzantine players are in addition to the n cost-minimizing ones.

Order of updates. Most of our lower bounds hold even if the players must update in some specified order, e.g. round-robin (i.e. cyclically) or the player to update is the one with the largest reduction in (perceived) cost [18, 28]. Our upper bounds on PoU hold no matter which player updates at any given step.

[5]When considering Byzantine players we no longer allow for perturbations in the costs perceived by players

[6]To avoid notation clutter, we omit parameters from the PoU and PoB notation when they are clear from context.

2.4 Consensus Games

In this section, we provide upper and lower bounds regarding the effect of uncertainty on consensus games which are asymptotically tight up to constant factors. In the uncertainty model these hold when ϵ is constant, and this holds in the Byzantine model for*any* number B of Byzantine players. Our bounds highlight a strong snowball effect of uncertainty in this class of games.

Consensus games [29] encapsulate a very common strategic interaction: there are two colors (or strategies), white (W) and red (R), which each player may choose; hence IR and BR dynamics coincide. Each player occupies a different vertex in an undirected graph G with vertices $\{1, \ldots, n\}$ and edges $E(G)$ (without self-loops). A player's cost is defined as the number of neighbors with the other color. We call an edge *good* if its endpoints have the same color and *bad* if they have different colors. The social cost is the number of bad edges (i.e. half the sum of all players' costs) plus one, which coincides with the game's potential. In the absence of uncertainty, the social cost decreases when a player decreases its own cost. Thus $PoU(0) = PoB(0) = 1$. Since the social cost is in $[1, n^2]$, $PoU(\epsilon) = O(n^2), \forall \epsilon$ and $PoB(B) = O(n^2), \forall B$.

2.4.1 Lower Bound and Upper Bound for Perturbation Model

Perturbation model. Since each action's cost is the number of neighbors of the other color, the perturbation model is as follows: if a vertex i has n' neighbors of some color, then a perturbation may cause i to perceive instead an arbitrary value in $[\frac{1}{1+\epsilon}n', (1+\epsilon)n']$.[7] In this model, only an $\epsilon = \Omega(\frac{1}{n})$ effectively introduces uncertainty.[8] We also assume $\epsilon \leq 1$ in this section.

We first provide a PoU *upper bound* for consensus games that depends on ϵ. It implies that the existing $\Theta(n^2)$ lower bound cannot be replicated for any $\epsilon = o(1)$. The proof is based on comparing the numbers of good and bad edges at the first move that increases the

[7]It may be natural to assume that the players perceive their number of neighbors as integers. Our results hold in this model as well.

[8]If $\epsilon < 1/3n$, consider a node with r red neighbors, w white neighbors, and $r > w$. Since r and w are both integers, $r \geq w + 1$. For a potential increasing move to occur, we must have $(1+\epsilon)^2 w \geq r \geq w + 1$. This implies that $n \geq w \geq 1/(2\epsilon + \epsilon^2) \geq 1/3\epsilon > n$.

social cost.

Theorem 1. *$PoU(\epsilon, consensus) = O(n^2\epsilon)$.*

Proof. We note that if no move is cost-increasing (i.e. any move would be valid for $\epsilon = 0$), then the potential function decreases. As potential and cost are equal, the social cost cannot increase in the absence of uncertainty.

Consider a strictly cost-increasing move by a player i. Let $m_b < m_g$ be the number of bad, respectively good, edges adjacent to i before i's move (their roles switch afterwards). Since i's move is cost-decreasing in the perturbed dynamics, $\frac{1}{(1+\epsilon)^2} \leq \frac{m_b}{m_g} < 1$. We get $m_b(1+\epsilon)^2 \geq m_g \geq m_b + 1$ i.e.

$$m_b \geq \tfrac{1}{2\epsilon+\epsilon^2} \tag{1}$$

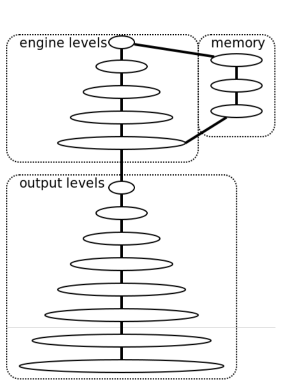
Thus before any cost-increasing move, in particular the first one, there must be $\Omega(\frac{1}{\epsilon})$ bad edges in the graph. As there are $O(n^2)$ edges in all, we get $PoU = O(n^2\epsilon)$, as desired. □

We now turn our attention to a lower bound. The proof is similar in spirit to a previous construction for weighted graphs due to Balcan et. al [12]. In that construction, there was a line of nodes where the weight between each successive pair of nodes grew exponentially. Our construction is similar in spirit, but works with unweighted graphs. There are two key ideas in our construction. First, rather than nodes we have a line of 'levels' which grow exponentially, and which can have alternating colors. Using this we can construct a graph with $\Omega(n^2\epsilon)$ edges, and find a state where most of the edges are bad. Secondly, rather than alternating every level, our construction alternates in blocks of two. This provides resilience which allows the adversary to maintain the construction despite allowing arbitrary orderings of player updates.

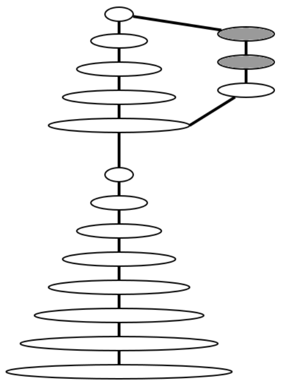
Theorem 2. *$PoU(\epsilon, consensus) = \Omega(n^2\epsilon^3)$ for $\epsilon = \Omega(\sqrt{\frac{1}{n}})$ even for arbitrary orderings of player updates.*

Proof. Assume for convenience $\frac{1}{\epsilon}$ is an integer.

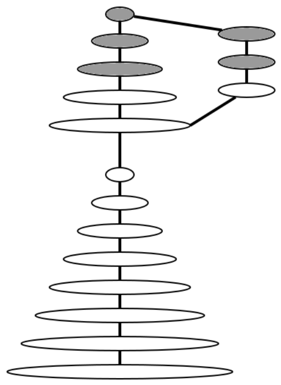
We first describe our construction assuming that the adversary can choose which player updates at any step. We describe in order the structure of the game, the initial and final states,



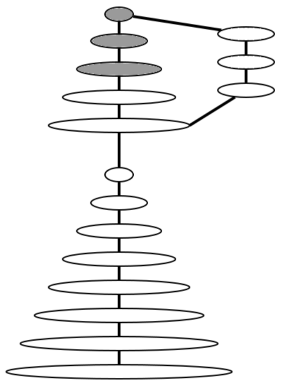
(a) The construction relies on three separate pieces, shown here.



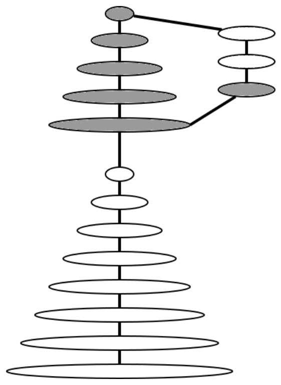
(b) Memory nodes start out with the opposite color to engine and output nodes.



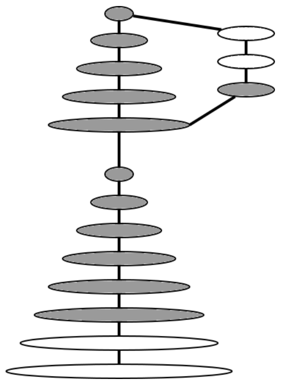
(c) Before the last two engine levels change color, the memory nodes switch color.



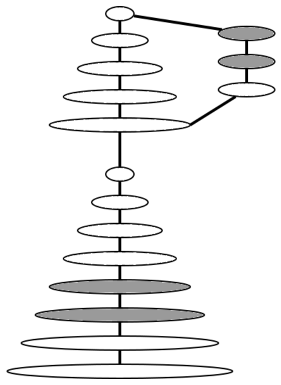
(d) The rest of the engine switches color in successive levels. Once the last engine level changes color, the first output level can change.



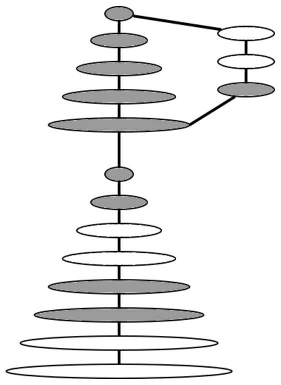
(e) The output levels change color in succession . . .



(f) . . . but the last two levels are 'fixed'.



(g) Each epoch switches the color of all nodes in the engine and memory, and fixes two more levels of output.



(h) Note that this final configuration is stable in the sense that for every node there is a perturbation that prevents it from changing color.

Figure 1: Construction for the $PoU(\epsilon) = \Omega(n^2\epsilon^3)$ (for $\epsilon = \Omega(\frac{1}{\sqrt{n}})$) lower bound in consensus games. In the diagram, levels are represented by nodes, and an edge between two levels implies that there is an edge between each pair of nodes on those levels. Note that engine and memory levels can take the color of the smaller level 'above' them.

calculate the Price of Uncertainty, and finally describe a schedule of the dynamics which lead from the initial state to the final state. We subsequently show that the adversary can reduce an arbitrary ordering (that he cannot control) to his desired schedule by compelling any player other than the next one in his schedule not to move.

Let us describe the structure of our lower bound. We construct a graph G (illustrated in Fig. 1 with $k_{eng} = 5, k_{out} = 8$) with

- $k_{eng} = \frac{1}{\epsilon}$ levels of "engine" nodes, where level i has $\frac{(1+\epsilon)^i}{\epsilon}$ nodes, for $i = 0 \dots k_{eng}$. Let n_{eng} denote the number of nodes in all engine levels, and note $n_{eng} \approx \frac{1}{\epsilon^2}$. Each node in an inner engine level is connected to each node in the two adjacent levels and not connected to any other engine node.

- k_{out} levels of "output" nodes, where level $i = 0 \dots k_{out}$ has $\frac{(1+\epsilon)^i}{\epsilon}$ nodes. Each node in an inner output level is connected to each node in the two adjacent levels and not connected to any other output node. We note that $k_{out} = \frac{1}{\epsilon}[\ln(n\epsilon^2) - \ln 2]$ as we describe shortly. Let n_{out} denote the number of nodes in all output levels, and note that $n_{out} \approx n^2\epsilon$.

- 3 levels of "memory" nodes, where each level has $\frac{1+\epsilon}{\epsilon}$ nodes. The first memory level is fully connected to the first engine level and to the middle memory level. We call this the *out-layer.* The third memory level is fully connected to the last engine level and to the middle memory level. We call this the *in-layer.* The middle layer is only connected to the other memory levels. We call this level the *storage.* Let n_{mem} denote the number of nodes in the memory levels.

There are no other edges in the graph. Let $n_{constr} = n_{eng} + n_{out} + n_{mem}$. If $n_{constr} < n$, then we also have $n - n_{constr}$ excess nodes not connected to anything. There are no other nodes in the graph.

It is easy to verify that we have at most n nodes in our graph.

$$\begin{aligned}
n_{eng} + n_{out} + n_{mem} &= \sum_{i=1}^{k_{eng}} \frac{(1+\epsilon)^i}{\epsilon} + \sum_{i=1}^{k_{out}} \frac{(1+\epsilon)^i}{\epsilon} + 3\frac{1+\epsilon}{\epsilon} \\
&= (1+\epsilon)\frac{(1+\epsilon)^{k_{eng}} - 1}{\epsilon^2} + (1+\epsilon)\frac{(1+\epsilon)^{k_{out}} - 1}{\epsilon^2} + 3\frac{1+\epsilon}{\epsilon} \\
&= (1+\epsilon)\frac{(1+\epsilon)^{1/\epsilon} - 1}{\epsilon^2} + (1+\epsilon)\frac{(1+\epsilon)^{1/\epsilon[\ln(n\epsilon^2) - \ln 2]} - 1}{\epsilon^2} + 3\frac{1+\epsilon}{\epsilon} \\
&\le (1+\epsilon)\frac{e-1}{\epsilon^2} + (1+\epsilon)\frac{\frac{1}{2}(n\epsilon^2) - 1}{\epsilon^2} + 3\frac{1+\epsilon}{\epsilon} \\
&\le (1+\epsilon)\frac{e-1}{\epsilon^2} + (1+\epsilon)\frac{\frac{1}{2}(n\epsilon^2) - 1}{\epsilon^2} + 3\frac{1+\epsilon}{\epsilon} \\
&\le (1+\epsilon)\left[\frac{n}{3} + \frac{n}{2}\epsilon^2 + 3\frac{1}{\epsilon}\right] \\
&\le n
\end{aligned}$$

Note that $(1+\epsilon)\frac{e-1}{\epsilon^2} > \frac{n}{3}$ provided that $\epsilon > \frac{2}{\sqrt{n}}$. By requiring $\epsilon \ge n^{-1/2}$ we ensure that engine and memory components are most a constant fraction of the nodes.

To prove our bound we will consider an initial state where all the storage and out-layer memory nodes are colored red, and all other nodes are white. As we shall show, we can reach from this a final state which alternates colors every other output level. (The color of the engine levels and memory levels will be the same as in the initial configuration.)

This immediately implies a lower bound on the price of uncertainty. We see that the initial number of bad edges is $\Theta(\frac{1}{\epsilon^2})$, (namely the number of bad edges between the out-layer of memory and the first engine level and the number of edges between the storage level and the in-layer, all of which have $\Theta(\frac{1}{\epsilon})$ nodes). The final number of bad edges is at least the number of bad edges between output levels. Note that this occurs either from odd levels to even levels, or the other way around (depending on the parity of k_{out}). Without loss of generality, this is at least

$$\sum_{i=1}^{(k_{out}-1)/2} \frac{(1+\epsilon)^{(2i)+(2i+1)}}{\epsilon\cdot\epsilon} = \frac{(1+\epsilon)}{\epsilon^2}\frac{(1+\epsilon)^{2k_{out}+2}-1}{(1+\epsilon)^4-1} = \Theta(\frac{(n\epsilon^2)^2}{\epsilon^3}) = \Theta(n^2\epsilon)$$

where we used that $(1+\epsilon)^{k_{out}} = \Theta(n\epsilon^2)$. Thus $PoU(\epsilon, consensus) = \Omega(n^2\epsilon^3)$.

We now show how it is possible to reach our desired final state from the initial state. We first do this by composing a schedule of updates for the players and perturbations at each step which lead the dynamics from the initial state to the final state. We then subsequently relax this and show that the schedule can be maintained even if the order in which players update is arbitrary. Note that every node is connected to all of the nodes in precisely two other levels (with the exception of the last level of nodes in both engine and output). Furthermore, the levels grow slowly enough so that a perturbation can prevent a node from telling which level that it is connected to is larger. If all the nodes in a level change color together, then we can treat the levels as supernodes with weighted edges between them that are growing exponentially.

Additionally, causing these levels to alternate colors improves the construction further. The engine and memory levels facilitate changing the color of the first output layer. There are $k_{out}/2$ epochs in the schedule. In epoch i, the engine and the first $k_{out}+1-2i$ output levels will match the memory color (that alternates between epoch) at the start of the epoch. The schedule ends when the output levels alternate color (in blocks of 2).

In order to describe the schedule, we must first introduce some notation. In all the relevant states, all nodes in each level of all components (i.e. engine, memory, output) will have the same color so we will use just one letter (W/R) to represent the color of all nodes in a level. The engine and output component will be represented as a string of k_{eng} and k_{out} letters respectively, and the memory component will be represented by three letters. The relevant colorings of engine and output levels are:

Engine. First j levels of one color, then $k_{eng}-j$ levels of the other: $\mathrm{R}^j\mathrm{W}^{k_{eng}-j}$ or $\mathrm{W}^j\mathrm{R}^{k_{eng}-j}$ for $0 \le j \le k_{eng}$.

Output. First i levels of one color, next j levels with the other, and the remaining levels with alternating colors in blocks of two: $\mathrm{R}^i\mathrm{W}^j(\mathrm{RRWW})^{(k_{out}-i-j)/4}$ and $\mathrm{W}^i\mathrm{R}^j(\mathrm{RRWW})^{(k_{out}-i-j)/4}$ for $0 \le i, j \le k_{out}$ with $k_{out} \ge i+j$. Note that we specify white for the last two output levels, red for the next to last pair of levels, then white for the previous pair etc.

A state is encoded by a triple, e.g. $(\mathrm{W}^{k_{eng}}, \mathrm{RRW}, \mathrm{W}^{k_{out}})$, for colors of the engine, memory and output nodes.

We now specify the schedule through which the initial state $(\mathrm{W}^{k_{eng}}, \mathrm{RRW}, \mathrm{W}^{k_{out}})$, in which all nodes are white except for the storage and out-layer memory nodes (see Fig. 1b), reaches the final state $(\cdot, \cdot, (\text{RRWW})^{k_{out}/2})$ in which the output levels have alternating colors in blocks of 2 (see Figure 1h). Note that any level in the engine or output can adopt the color of the level before it by a $1+\epsilon$ perturbation.

Consider without loss of generality if level $i-1$ is red. We claim that a node in level i can change color to red. Since all if its neighbors in level $i-1$ are red, then this node has at least $\frac{1}{\epsilon}(1+\epsilon)^{i-1}$ red neighbors. An ϵ-perturbation can cause the node to perceive this as $\frac{1}{\epsilon}(1+\epsilon)^{i}$. This node also has at most $\frac{1}{\epsilon}(1+\epsilon)^{i+1}$ white neighbors, as the nodes on level $i+1$ are the only possible neighbors. Even if all of these are colored white, an ϵ-perturbation may cause the node to perceive this also as $\frac{1}{\epsilon}(1+\epsilon)^{i}$. The node is therefore indifferent between colors red and white, and can break ties arbitrairly. Thus this particular node on level i, and similarly all nodes on level i, can update to take the color red.

The schedule is indicated by Figure 1. Figure 1b-1f show the schedule within a particular epoch, and the remaining figures show how the epochs reach the desired ending state. Without loss of generality we may assume this is epoch which starts with the engine being white. For epoch i, this is configuration $(\text{W}^{k_{eng}}, \text{RRW}, \text{W}^{k_{out}-2i}(\text{RRWW})^{i/2})$.

First the engine takes the color of the memory nodes. The first engine level takes the color of the out-layer of memory nodes, and then the first $k_{eng}-2$ engine levels change their color in order. This is depicted in 1c. This configuration $(\text{RW}^{k_{eng}-1}, \text{RRW}, \text{W}^{k_{out}-2i}(\text{RRWW})^{i/2}i) \to \cdots \to (\text{R}^{k_{eng}-2}\text{W}^2, \text{RRW}, \text{W}^{k_{out}-2i}(\text{RRWW})^{i/2})$. Then each node in the storage level changes to the color of the in-layer, and then the out-layer adopts the color of storage. Finally the last two engine levels change color. $(\text{R}^{k_{eng}-2}\text{W}^2, \text{RRW}, \text{W}^{k_{out}-2i}(\text{RRWW})^{i/2}) \to (\text{R}^{k_{eng}-2}\text{W}^2, \text{WWW}, \text{W}^{k_{out}-2i}(\text{RRWW})^{i/2}) \to (\text{R}^{k_{eng}}, \text{WWR}, \text{W}^{k_{out}-2i}(\text{RRWW})^{i/2})$ At this point, all non-output nodes have reversed color, as shown in 1e.

Next the output levels will change color, as in 1f. The first output level is connected to more than twice as many nodes of last engine level than to the next level of output, so these nodes change color to state $(\text{R}^{k_{eng}}, \text{WWR}, \text{RW}^{k_{out}-2i-1}(\text{RRWW})^{i/2})$. Thereafter, until level $k_{out}-2i$, each output level adopts this color reaching state $(\text{R}^{k_{eng}}, \text{WWR}, \text{R}^{k_{out}-2i-1}\text{WW}(\text{RRWW})^{i/2})$.

Finally epoch i is complete, and this is the starting configuration for epoch $i+1$.

Arbitrary Orderings. Finally we note that the adversary can use this schedule, even if he does not have control over who updates at each time step. We do this by showing that given some arbitrary ordering, he can reduce it to his desired schedule by compelling players not to move. In each epoch, there are two types of nodes: those the adversary wishes to not change color (the nodes already fixed) and those the scheduled to change color precisely once.

Consider the nodes scheduled not to change color. Each one of these is in the final k levels of output. Consider such a node. Without loss of generality, it has a number of red neighbors equal in size to the previous level and a number of white neighbors equal to the size of the next level. By overestimating the number of red neighbors by a factor of $(1 + \epsilon)$ and underestimating the number of white neighbors by the same, this node perceives equal costs for either red or white, and so can stay with the same color they have currently chosen. The only fixed nodes that do not have red level and a white level adjacent are the nodes in the first fixed level — but these have an excess of neighbors that are their current color. So these can preserve their color throughout the epoch.

Now consider an odd numbered epoch. Of the remaining, non-fixed nodes, memory nodes in the storage and out-layer levels wish to change color from red to white; all other nodes wish to change from white to red. (The colors are reversed for even numbered epochs.) We now wish to show that for any node that has updated from white to red there exists a perturbation that keeps it red. Note that a perturbation exists to choose a particular color if at least a $\frac{1}{2} - \epsilon$ fraction of its neighbors are that color. For all non memory nodes, the number of red neighbors is monotonically increasing over the course of the epoch, so these nodes can stay red. (The sole exception to this is the first level of engine nodes, but note that the memory nodes it is connected to change color to white only after it and its successor level have changed to red. This is enough to prevent these nodes from changing back to white.)

Next we consider the storage and out-layer levels of memory nodes. The adversary will cause them to change color before any nodes in the last two engine levels change color, but after the entire first and second engine levels have changed color. Note that the adversary can prevent them from updating: the storage level has an equal number of red neighbors and white neighbors until all the in-layer has changed color, and nodes in the out-layer only have red neighbors. Further, the in-layer cannot change color until the last engine level has changed color. Thus these nodes will not change the color of these nodes until the after the second engine level changes color.

Since the storage level has an equal number of neighbors of each color, once the second engine level has changed color, the storage can be compelled to update from white to red. After all nodes in the storage level have changed color, the nodes in the out-layer can update. These nodes have an equal number of neighbors of each color, and so can stay the same color throughout the rest of the epoch. The nodes in the storage level have at least as many white neighbors as red neighbors even after the in-layer updates, so they can also decline to change color for the rest of the epoch.

We note that the penultimate engine level can decline to change color, since, until it changes color, it has more white neighbors than red neighbors.

Finally, consider the in-layer of memory nodes. Until the nodes in the last engine level change color, these have only white neighbors, so they will not change color. After the storage nodes have changed color, the number of red neighbors is monotonically increasing. Once all the nodes in the last engine level have changed color, they have more red neighbors than white neighbors and so cannot change color to white.

Thus we have shown how for every node at every timestep there is a perturbation which preserves the adversary's schedule. □

We note that the previously known lower bound of $\Omega(1 + n\epsilon)$ due to Balcan et. al [12] was based on a much simpler construction. When $\epsilon = \sqrt{\frac{1}{n}}$, the bounds meet and for larger values of ϵ our new results can be drastically better. Our new bound is better by a factor of at least $n^{1/3}$ for $\epsilon = \Omega(n^{-1/3})$, and when ϵ is a constant the bound is improved by a factor of n. We also note that since $PoU(\epsilon) = O(n^2)$ for any ϵ, it implies a tight PoU bound of $\Theta(n^2)$ for *any* constant ϵ.

2.4.2 Tight Bound for Byzantine Players

As described earlier, Byzantine players can choose their color ignoring their neighbors' colors (and therefore their own cost). Note however the Byzantine players cannot alter the graph[9]. We show a tight bound on the effect of B Byzantine players, for any B: the effect of one Byzantine player is very high, of order $n\sqrt{n}$ and that the subsequent effect of $B \leq n$ Byzantine players is proportional to the square root of B. As was the case for PoU, the effect of uncertainty is decomposed multiplicatively into a power of n and a power of the extent of uncertainty (ϵ for PoU, B for PoB).

Theorem 3. *$PoB(B, consensus) = \Theta(n\sqrt{n \cdot B})$.*

The rest of this section establishes this tight bound. We begin by providing a high-level overview of the proof.

- We first consider an instance of a consensus game under Byzantine uncertainty. We show that this instance can increase its cost to a value of k (i.e. $\exists S$ such that $cost(S) = k$) then (by Lemma 1) there exists a B-flippable graph $F_{G,k}$ (we shall define flippable shortly), such that $|V(F_{G,k})| \leq 3n$ and $|E(F_{G,k})| \geq k$.
- We then seek to bound the number of edges in flippable graphs. We identify a particular B-flippable graph, $F_{seq}(n, B)$, on n vertices. Among B-flippable graphs F on n vertices, $F_{seq}(n, B)$ has the most edges. Thus $|E(F)| \overset{Lemma\ 2}{\leq} |E(F_{seq}(n, B))| \overset{Lemma\ 3}{=} \Theta(n\sqrt{nB})$.
- As $PoB(B, G) = k \leq |E(F_{G,k})|$, we have $PoB(B, G) \leq \max_S cost(S) \leq \max_{\text{flippable } F} |E(F)| = O(n\sqrt{nB})$.
- Finally, half of $F_{seq}(n, B)$'s edges can be made bad, i.e. $PoB(B, F_{seq}(n, B)) = \Omega(n\sqrt{nB})$.

The proof of the $O(n\sqrt{n \cdot B})$ upper bound follows from Lemmas 1, 2 and 3 below. The key to this bound is the notion of a flippable graph. For any consensus game, let S_{red} be the

[9]For the lower bound, we assume that a player will break ties in our favor when he chooses between two actions of equal cost. With one more Byzantine player the same bound holds even if players break ties in the worst possible way for us. For the upper bound, we assume worst possible players' moves from the social cost point of view.

configuration where all nodes are red, and similarly let S_{white} be the configuration where all nodes are white.

Definition 1 (B-Flippable graph)**.** *Consider graph G on n vertices of which B are designated special nodes and the other $n - B$ nodes are called normal. We say G is B-flippable (or just flippable when B is clear from context) if in the consensus game defined on G where the special nodes are the Byzantine agents, S_{white} is B-Byz-reachable S_{red}.*

We now describe the concept of a *conversion dynamics* in a consensus game which we use in several of our proofs. In such a dynamics, we start in a state where all vertices are red and have Byzantine players change their color to white. Then all normal nodes are allowed in a repeated round-robin fashion to update, so long as they are currently red. This ends when either every vertex is white or no vertex will update its color.

We note that in a flippable graph the conversion dynamics induces an ordering of the normal vertices: nodes are indexed by the order they first change color from red to white, with the first node switching to white being labelled 1, the next 2, and so on. We note that there may be more than one valid ordering. In the following with each B-flippable graph, we shall arbitrarily fix a canonical ordering (by running the conversion dynamics). Where there is sufficient context, we shall use v a vertex interchangeably with its index in this ordering. Using this ordering we induce a canonical orientation by orienting edge uv from u to v if and only if $u < v$. We also orient all edges away from the B special nodes. To simplify notation, we shall write $v_{in} = |\delta^{-}(v)|$ and $v_{out} = |\delta^{+}(v)|$ for a vertex v's in-degree and out-degree respectively. We note that by construction, for a flippable graph we have $v_{in} \geq v_{out}$. One can easily show the following:

Claim 1. *A graph is B-flippable if and only if there exists an ordering on $n - B$ normal vertices of the graph such that, in the canonical orientation of the edges such that for every normal vertex v, $v_{in} \geq v_{out}$. A graph is B-flippable if and only if for every pair states S, S', S is B-Byz-reachable from S'.*

Proof. We first show a graph is B-flippable if and only if there exists an ordering on $n - B$ normal vertices of the graph such that the canonical orientation of the edges such that for every normal vertex v, $v_{in} \in v_{out}$.

Suppose a graph has such an ordering. Let the adversary start it in the all red configuration, update each Byzantine player to white, and the update according to the specified ordering (breaking ties towards white). Observe that each vertex updates to white immediately. Thus the graph is B-flippable. On the other hand, consider a B-flippable graph. We note that the conversion ordering has our property, as desired.

We now show that a graph is B-flippable if and only if for every pair states S, S', S is B-Byz-reachable from S'. One implication is trivial, as S_{white} B-Byz-reachable from S_{red} is the definition of B-flippable. The other direction is also easy. We show this by inducting over flippable subgraphs. Order the nodes by the conversion dynamics, and let v be the last node. Note that $G \setminus \{v\}$ is also flippable, with the same conversion ordering. We first color v by either leaving it the proper color or by changing the color of all vertices of G. In either case, having colored v properly, we restrict our attention to the subgraph $G \setminus \{v\}$ and color it appropriately by induction. Note that since v is last in the conversion ordering, all other vertices can swap their color even if v is colored differently. $\square$

Lemma 1. *Fix a game G on n vertices, B of which are Byzantine, and a pair of configurations S_0 and S_T such that S_T is B-Byz-reachable from S_0. If $cost(S_0) \leq n$, then there exists a B-flippable graph F with at most $3n$ nodes and at least k edges (in total).*

Proof. Let our initial consensus game G have an initial configuration S_0 and a B-Byz-reachable configuration S_T. Recall that S_{red} refers to the state where every vertex is red, and likewise that S_{white} is the state where all vertices are white.

We shall first construct a consensus game G' such that there exists a configuration S'_T which is B-Byz-reachable configuration S_{red}, such that $cost(S'_T) \geq cost(S_T)$. Informally, this means that the intermediate game G' will have at least as many bad edges in the configuration S'_T as G does in S_T.

We then transform the game G' into a B-flippable graph G'' on the same vertex set by deleting edges of G' while ensuring that S_{white} is B-Byz-reachable from S_{red} in G'', and $|E(G'')| \geq cost(S'_T)$.

We now describe the structure of G'. Consider G in state S_0. For each edge $e = uv$ that is bad in S_0, introduce two nodes, a mirror node m_e and a helper node h_e. We delete edge e, and put in new edges um_e, $m_e v$ and $m_e h_e$. Additionally, the mirror and helper nodes have an edge from each Byzantine agent. Thus we introduce two nodes for each bad edge. By assumption there are at most n bad edges in S_0 and thus at most $2n$ new nodes overall.

Note that by controlling the order of updates and being able to choose how other nodes break ties, the adversary can control the color of the mirror node independently of the rest of the game. When attempting to change node m_e, to color i, the adversary first asks the Byzantine node to change color to i. Then the node h_e is chosen to update, and chooses to break its tie in favor of color i. Then the mirror node does the same.

We now show how the dynamics can reach a state S'_T with $k = cost(S_T)$ bad edges while starting from S_{red}.

Let $v_1, v_2, \ldots, v_t$ be the ordering of updates that reaches k bad edges, with associated configurations of G, S_1, S_2 and so on. Note that the v_i may not be distinct, but that they do not include the Byzantine vertex. Further, let I_r be the set of vertices that are red in S_0 and I_w be the set of vertices initially white. The dynamics proceeds in two stages. In the first stage, we only update vertices of I_r, in the second stage we only update vertices of I_w. Since the mirror nodes isolate set I_r from set I_w and vice-versa, the adversary can simulate the dynamics from S_0 to S_T on I_r. In the set I_w, the adversary simulates the dynamics from $\bar{S}_0$ to $\bar{S}_T$ — the dynamics with color of every node reversed.

Let S'_0 in G' be state where every vertex is colored red, i.e. S_{red}. Then, the Byzantine agent updates to white, and queries every helper node, and then every mirror node. This simulates the original dynamics in G to the vertices of I_r.

The adversary then uses the following ordering: For $i \in [1, t]$, v_i is updated if $v_i \in I_r$. Then for each mirror edge with $a \in I_r, b \in I_w$, the adversary induces m_{ab} to take color of b in S_{i+1}. This maintains the invariant that all nodes of I_r have identical numbers of red and white edges as in the state S_i of G when they make their updates, so each vertex of I_r updates the same as before.

Now we wish to update the nodes of I_w. For any state S of G let $\bar{S}$ denote the state in which the color of every vertex in G is flipped. The adversary will update the nodes of I_w following the dynamics from $\bar{S}_0$ to $\bar{S}_T$. Note that since the vertices of I_w are defined as those that have the color white in S_0, they all are red in $\bar{S}_0$. We perform the same update procedure for I_w as we did for I_r, ensuring that we get to $\bar{S}_T$. For $i \in [1,t]$, v_i is updated if $v_i \in I_w$. Then for each mirror edge with $a \in I_r, b \in I_w$, the adversary induces m_{ab} to take color of b in $\bar{S}_{i+1}$. After this procedure, the vertices of b are the same color as in $\bar{S}_T$.

Finally we update each helper edge to be the opposite color of the mirror edge. This final configuration is S'_T.

We finally note that all edges internal to I_r that were bad in S_T are bad in S'_T. Likewise, all edges internal to I_w are bad in S'_T if they were bad in S_T. The only bad edges of S_T that might not be present are those edges which have been replaced by mirror nodes. But for each of these edges there is a bad edge its mirror node to its helper node in S'_T, which suffices. Thus from S_{red} we reach a state S'_T with at least k bad edges.

We have constructed G' and a state S'_T such that $cost(S'_T) \geq cost(S_T)$ and S'_T is B-Byz-reachable from S_{red} in G'. We now find a set of edges in G' that we can delete creating G'' such that S_{white} is B-Byz-reachable from S_{red} in G'' and no edge which is bad in S'_T is deleted.

To construct G'' we must first examine the vertices of G'. Let R be the set of vertices in G' that never change color to white in the conversion ordering. Note that these vertices cannot change to white in *any* update ordering while performing IR. To see this, we observe that for all nodes $v \in R$, v must have more edges to nodes of R than nodes outside of R, else v would change color in the conversion dynamics. Assume to the contrary that some node v in R changes color. Without loss of generality we examine the first time in the dynamics that this happens. But note that all of v's inside R at this time are still red, so v does not change color after all.

We shall now describe the structure of G''. G'' will be identical to G' with internal edges of R deleted. Formally, $V(G'') = V(G')$ and $E(G'') = E(G') \setminus \{e = (a,b) : a,b \in R\}$. This does not change our PoB; each of the deleted edges had both endpoints red in every state of the dynamics, and hence were not bad edges in any state, and thus were not bad in S'_T.

We can follow the old dynamics ignoring any queries to nodes of R, and reach the final configuration S'_T.

Furthermore, every vertex can now be colored white. This is true as every node is only connected to vertices that can be colored white. If we use conversion dynamics, then all of these vertices are turned white, and the nodes of R are only connected to white nodes. Thus they can also turn white.

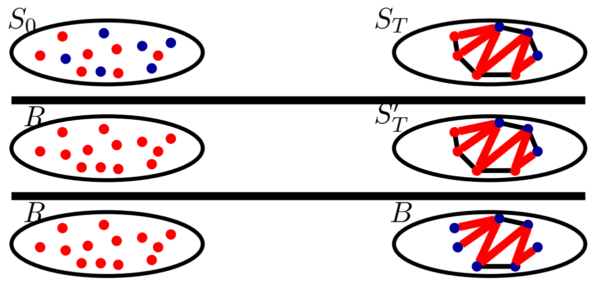
Thus, given G, we have constructed G'' where $E(G'') \geq cost(S_T)$, $|V(G'')| \leq 3n$, and S_{white} is B-Byz-reachable from S_{red} in G''. In short, G'' is a B-flippable graph with the desired parameters. $\square$

Definition 2 ($F_{seq}(n, B)$)**.** *Let $F_{seq}(n, B)$ be the B-flippable graph with $n - B$ normal nodes with labels $\{1, 2, \ldots, n - B\}$. There is an edge from each special node to each normal node. Every normal node v satisfies $v_{out} = \min(v_{in}, (n - B) - v)$, and v is connected to the nodes of $\{v + 1, \ldots, v + v_{out}\}$. This is called the* no-gap *property. In general, if $k = \min(v_{in}, n - v)$ then v has out-arc set $\{v + 1, \ldots, v + k\}$.*

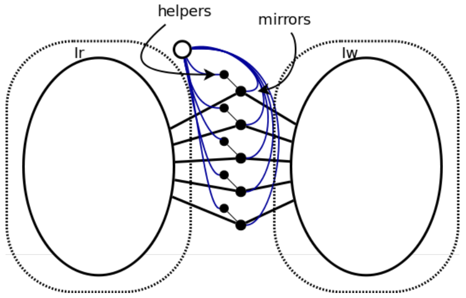
By claim 1 we immediately get that F_{seq} is B-flippable. Our upper bound follows by showing $|E(F)| \leq |E(F_{seq}(n, B))|$ for any flippable graph F on n vertices. For this, we take a generic flippable graph and transform it into F_{seq} without reducing the number of edges. We say there is a *gap*(a, b, c) for $a < b < c$ if vertex a does not have an edge to b but does have an edge to c, see Figure 3a. Note that this is defined in terms of an ordering on the vertices; we use the conversion ordering for each graph. Since we fix B and n, we use shorthand F_{seq} for $F_{seq}(n, B)$.

Lemma 2. *A B-flippable graph on n vertices has at most as many edges as $F_{seq}(n, B)$.*

Proof. For this proof we only consider B-flippable graphs on n vertices with B special nodes. We refer to the *minlex pair* of a graph as the lexicographically minimal pair (a, b) in a graph such that *gap*(a, b, c) is present in the graph for some c. This proof proceeds by backwards induction on graphs classified by their minlex pair.

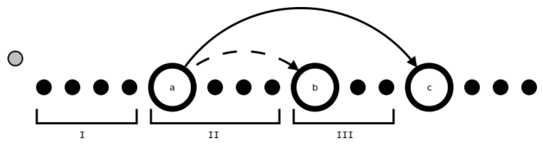


(a) First the game is converted to one in which the all red configuration can reach a state of social cost no less than $cost(S_T)$. Next, we show that we can reach the all white configuration without deleting any bad edges.

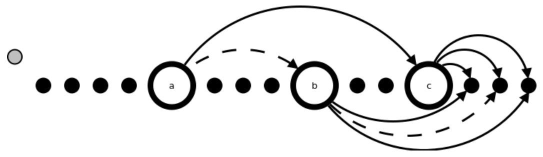


(b) The sets I_r and I_w are only connected by mirror nodes. The blue dotted edges connect the new mirror and helper nodes to the Byzantine player. The large solid dots are mirror nodes, while the smaller dots are helper nodes.

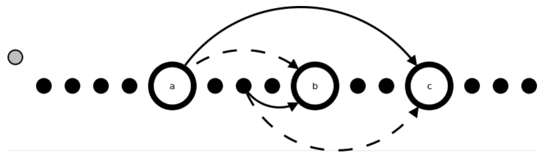
Figure 2: The diagram on the left shows the three stages of Lemma 1, while the diagram on the right shows a closer look at the first stage. Note that white nodes are drawn as blue in this image.



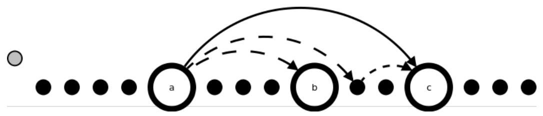
(a) $gap(a, b)$ is present in this graph. Note that block I does not include node a.



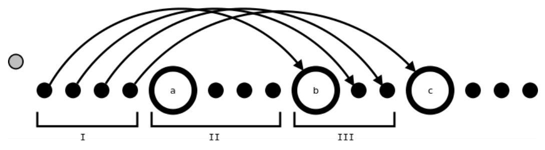
(b) When $b_{out} < c_{out}$



(c) $d \rightarrow b$ but $d \not\rightarrow c$ can be changed by adding $a \rightarrow b$ and $d \rightarrow c$ and deleting $a \rightarrow c$ and $d \rightarrow b$



(d) If $(iii) \not\rightarrow c$ then the graph is not edge maximal. We delete $a \rightarrow c$ and add in arcs $a \rightarrow (iii)$ and $(iii) \rightarrow c$



(e) Demonstrating that $c^I_{in} \geq b^I_{in} - x$

Figure 3: Relevant nodes and blocks for the proof of Lemma 2. The nodes are arranged from left to right by increasing index. Dashed lines represent arcs not present. Only one Byzantine player is pictured (as the gray node); its edges (one to each node) are omitted to avoid clutter. Figure 3a shows the relevant definitions, while the rest show relevant cases.

We first note that F_{seq} is precisely the (B-flippable) graph on n vertices with no gaps, and so it (vacuously) has a minlex pair greater than that of any other graph. This serves as our base case.

We now assume any graph with minlex pair (x, y) determined by $gap(x, y, z)$ with $(x, y) >_{lex} (a, b)$ has at most as many edges as F_{seq}.

Among B-flippable graphs with minlex pair (a, b) consider the graph G with the most edges. Let G's lexicographically minimal gap be $gap(a, b, c)$. We shall show that $|E(G)| \leq |E(F_{seq})|$ which implies that all graphs with minlex pair $gap(a, b, \cdot)$ have fewer edges than F_{seq}.

If $c_{out} < c_{in}$, we remove $a \to c$ and add $a \to b$. This results in a graph, G', with the same number of edges, but with a strictly later minlex pair. Note that G' is also B-flippable, since we maintain the propery that for all v, $v_{in} \geq v_{out}$. By our inductive hypothesis, G' has fewer edges than F_{seq}, and so the induction holds.

If $b_{out} < c_{out}$, find d with $c \to d$ and $b \not\to d$, as shown in Figure 3b. Create G' by removing edges $a \to c$ and $c \to d$ while adding edges $a \to b$ and $b \to d$. G' maintains the condition on vertices, and is thus a B-flippable graph with the same number of edges, but with a later minlex pair. Thus by our induction G' has fewer edges than F_{seq}.

Consider any d with $a < d < b$. If $d \to b$ but $d \not\to c$, we remove edges $d \to b$, $a \to c$ and add edges $d \to c$ and $a \to b$, as shown in Figure 3c. This results in a B-flippable graph with the same number of edges but a strictly greater minlex pair. Thus again by induction, it has fewer than F_{seq} edges.

Now assume for the sake of contradiction that none of the above hold. Thus $b_{out} \geq c_{out} = c_{in}$. We show $c_{in} > b_{in}$ which leads to a contradiction as $b_{in} \geq b_{out} \geq c_{out} = c_{in}$.

Let block I be the nodes of index $< a$, block II be the nodes with index in $[a, b)$, and block III be the nodes with index in $[b, c)$ — see the figure below. We will count the vertices with edges into both b and c in each block. We use v^J_{in} to denote the number of edges to v in block J, for $J \in \{I, II, III\}$.

We begin with block II. Let (ii) be a vertex in block II. By assumption, for any vertex (ii) such that $(ii) \to b$, we must also have $(ii) \to c$. So $c^{II}_{in} \geq b^{II}_{in}$. Furthermore, a is in this block, so $c_{in}^{II} \geq b_{in}^{II} + 1$.

There are x vertices in block III (including b). These must all go to c. Otherwise, suppose $(iii) \not\to c$ where (iii) is a vertex in block III, as shown in Figure 3d Note that $a \not\to (iii)$, or else the minimal gap is actually $gap(a, b, (iii))$. We delete edge $a \to c$ and add edges $a \to (iii)$ and $(iii) \to c$. This is a graph with more edges but with $gap(a, b, (iii))$. This violates our choosing the graph G with the most edges. Thus $c_{in}^{III} = x$. Further, none of the vertices in block III go to b as they all have indices exceeding b, so $b_{in}^{III} = 0$.

There are b_{in}^{I} nodes in block I which go to b. We note since there are no gaps before a, each of these nodes $i < a$ hit their next i_{in} neighbors.

Also note that each successive node has at least as many in-edges (and therefore out edges) as its predecessor. This is because the graph is $F_{seq}(a, B)$ before this point as there are no gaps prior to a. So of the b_{in}^{I} nodes that hit b, 1) they are consecutive, and end at a, 2) at least $b_{in}^{I} - x$ hit c. We get $c_{in}^{I} \geq b_{in}^{I} - x$. This is shown in Figure 3e.

Summing overall we find that $c_{in} = c_{in}^{I} + c_{in}^{II} + c_{in}^{III} \geq (b_{in}^{I} - x) + (b_{in}^{II} + 1) + x = b_{in} + 1 > b_{in}$.

Thus we have reached the desired contradiction. $\square$

Our last lemma tightly counts the number of edges in $F_{seq}(n, B)$ via an inductive argument and thus, by Lemma 2, it also upper bounds the number of edges in any flippable graph.

In the following discussion, it will be helpful to refer to the node of a given index. To this end, we shall write $\underline{x}$ to represent the node of index x. Similarly, we can write $\underline{x}_{in}$ to represent its number of in-arcs.

Lemma 3. *If $B \leq \frac{n}{2}$, the flippable graph $F_{seq}(n, B)$ has $\Theta(n\sqrt{nB})$ edges.*

Proof. Note that the number of in-edges at a given node is monotonically increasing. (This follows as the no-gap property ensures that there is at most one node which has an edge to to $\underline{i}$ but not $\underline{i+1}$, and $\underline{i}$ itself is a node which contributes to $\underline{i+1}$.) Let $f(k)$ be the lowest-indexed node in $F_{seq}(n, B)$ which has k in-edges (not counting edges from the Byzantine player). We show by induction that, in a graph with B Byzantine players, $f(jB) = \binom{j+1}{2}B + 1$. Our base case is trivial, as $\underline{1}$ has no in-edges ($f(0) = 1$).

Now suppose that $f(jB) = \binom{j+1}{2}B+1$, and we wish to show that $f((j+1)B) = \binom{j+2}{2}B+1$.

We first count the number of in-edges of $\underline{\binom{j+2}{2}B+1}$. Note that $\underline{\binom{j+1}{2}B+1}$ has at least jB in-edges by our induction, so it must have $(j+1)B$ out-edges. This implies that it has an out-edge to $\underline{\left(\binom{j+1}{2}B+1\right)+(j+1)B} = \underline{\binom{j+2}{2}B+1}$.

Thus, by the no-gaps property which defines F_{seq}, $\underline{\binom{j+2}{2}B+1}$ has in-edges from all nodes in the range $\left[\binom{j+1}{2}B+1, \binom{j+2}{2}B\right] = (j+1)B$ in-edges, as desired. Thus $f((j+1)B) \leq \binom{j+2}{2}B+1$, as desired.

To show $f((j+1)B) \geq \binom{j+2}{2}B+1$, we consider $\underline{\binom{j+2}{2}B}$, and compute how many in-edges it has. Note that $\underline{\binom{j+1}{2}B}$ has fewer than jB in-edges, so it has strictly fewer than $(j+1)B$ out-edges. The last node it reaches has index at most $\binom{j+2}{2}B-1$. Thus the node has in-edges from nodes with indices in the range $\left(\binom{j+1}{2}B, \binom{j+2}{2}B\right)$, and thus has fewer than $(j+1)B$ in-edges. Thus as desired we have shown that $f((j+1)B) \geq \binom{j+2}{2}B+1$.

Together these imply that $f((j+1)B) = \binom{j+2}{2}B$, and our induction holds. Now we use this to approximate the number of edges in $F_{seq}(n,B)$.

We wish to compute $\sum_{x=1}^{n-B} \underline{x}_{in}$. To do so we shall compute this as a function holding B fixed: $\sum_{x=1}^{m} \underline{x}_{in}$. (This is the number of edges in $F_{seq}(m+B,B)$). We note that when $x \in [\binom{j}{2}B+1, \binom{j+1}{2}B]$, that $(j-1)B \leq \underline{x}_{in} < jB$.

We break our sum into the following sub-sums

$$\sum_{x=1}^{m} \underline{x}_{in} = \sum_{j=1}^{\sqrt{\frac{m}{B}}} \sum_{x=\binom{j}{2}B+1}^{\binom{j+1}{2}B} \underline{x}_{in} \leq \sum_{j=1}^{\sqrt{\frac{m}{B}}} \sum_{x=\binom{j}{2}B+1}^{\binom{j+1}{2}B} jB = \sum_{j=1}^{\sqrt{\frac{m}{B}}} (j+1)BjB = O(m\sqrt{mB}).$$

We have just shown that $F_{seq}(m+B,B)$ has $O(m\sqrt{mB})$ edges. Similarly, we may bound this from below.

$$\sum_{x=1}^{m} \underline{x}_{in} = \sum_{j=1}^{\sqrt{\frac{m}{B}}} \sum_{x=\binom{j}{2}B+1}^{\binom{j+1}{2}B} \underline{x}_{in} \geq \sum_{j=1}^{\sqrt{\frac{m}{B}}} \sum_{x=\binom{j}{2}B+1}^{\binom{j+1}{2}B} (j-1)B = \sum_{j=1}^{\sqrt{\frac{m}{B}}} (j+1)B(j-1)B = \Omega(m\sqrt{mB}).$$

By plugging in $m = n-B$, we see that $|E(F_{seq}(n,B))| = \Theta\left((n-B)\sqrt{(n-B)B}\right)$. Note however that $n/2 \leq n-B \leq n$. Since the number of normal nodes is $n-B = \Theta(n)$, we conclude that $F_{seq}(n,B)$ has $\Theta(n\sqrt{nB})$ edges, as desired. $\square$

Proof. We first argue that the $PoB(B, consensus) = O(n\sqrt{nB})$. Consider a consensus graph G on n nodes, and a pair of configurations S_0 and S_T B-Byz-reachable from S_0. If $B \geq n/2$, then the statement is trivial, so we may assume that $B < n/2$. We assume $cost(S_0) < n$: if $cost(S_0) \geq n$, since G has fewer than n^2 edges, we get $PoB(B, G) \leq n^2/n = n$. Denote by $k := cost(S_T) - 1$ the number of bad edges in S_T. By Lemma 1, we demonstrate a flippable graph F on fewer than $3n$ nodes, with at least k edges. By Lemma 2, F has at most as many edges as $F_{seq}(3n, B)$, which has only $O(n\sqrt{nB})$ edges by Lemma 3. We get $PoB(B) = O(n\sqrt{nB})$.

It will now be enough to prove that $PoB(B, F_{seq}(n, B)) = \Omega(n\sqrt{nB})$. We claim now that if G is a flippable graph with m edges, then $PoB(B, G) \geq \frac{m}{2}$. We get this via the following probabilistic argument using the fact that the adversary can color G arbitrarily (by claim 1). Consider a random coloring of the graph, where each node is colored white independently with probability 1/2. The probability an edge is bad is 1/2, so in expectation, there are $m/2$ bad edges. Thus some state has at least $m/2$ bad edges and it is reachable via dynamics from any other state (claim 1) since G is a flippable graph. This establishes $PoB(B, G) \geq \frac{m}{2}$. Since $F_{seq}(n, B)$ is flippable and it has $m = \Theta(n\sqrt{nB})$ edges, we get $PoB(B, F_{seq}(n, B)) = \Omega(n\sqrt{nB})$. □

In contrast to the existing bound $PoB(1) = \Omega(n)$, our bound is parametrized by B, sharper (by a $\Theta(\sqrt{n})$ factor for $B = 1$) and asymptotically tight.

2.5 Set-Covering Games and Extensions

Set-covering games (SCG) [26, 37] are a basic model for fair division of costs, and have wide applicability, ranging e.g. from a rental car to advanced military equipment shared by allied combatants. A set-covering game is played over m sets where each set j has weight w_j. Each player i uses exactly one set from $\{1 \ldots m\}$ (some sets may not be available to i). All users of a set share its weight fairly: letting $n_j(S)$ denote the number of users of set j in state S, each such player has cost $\frac{w_j}{n_j(S)}$. This game admits the potential function

$$\Phi(S) = \sum_{j=1}^{m} \sum_{i=1}^{n_j(S)} \frac{w_j}{i} = \sum_{j=1}^{m} \Phi^j(S) \text{ where } \Phi^j(S) = \sum_{i=1}^{n_j(S)} \frac{w_j}{i} \quad (2)$$

$\Phi^j(S)$ has a simple, intuitive representation: it can be viewed as a stack (see Fig. 5a and 5b) of $n_j(S)$ *chips*, where the i-th chip from the bottom has a cost of w_j/i. When a player i moves from set j to j' one can simply move the topmost chip for set j to the top of stack j'. This tracks the change in i's costs, which equals by definition the change in potential Φ. We will only retain the global state (number of players using each set) and discard player identities.[10] This representation has been introduced for an existing PoU_{IR} upper bound; we refine it for our improved upper bound.

SCGs have quite a small gap between potential and cost [2]: $cost(S) \leq \Phi(S) \leq cost(S)\Theta(\log n), \forall S$. Hence without uncertainty, the social cost can only increase by a logarithmic factor: $PoU(0) = PoB(0) = \Theta(\log n)$.

2.5.1 Upper Bound for Improved-Response

We start with an upper bound on PoU_{IR} in set-covering games that only depends on the number m of sets. This is a relevant quantity as we expect it to be less than the number n of players (since players share set weights). In particular for $\epsilon = O(\frac{1}{m^2})$ we obtain a logarithmic $PoU_{IR}(\epsilon)$. That is, such an ϵ (magnitude of uncertainty) has virtually no effect on set-covering games since $PoU_{IR}(0)$ is also logarithmic. We leverage our bound's independence of n in Section 2.5.1.1 below and extend it to classes of matroid congestion games.

Theorem 4. *$PoU_{IR}(\epsilon, \text{set-covering}) = (1+\epsilon)^{O(m^2)}O(\log m)$ for $\epsilon = O(\frac{1}{m})$.*

Proof. We let J_0 denote the sets initially occupied and $W_0 = cost(S_0) = \sum_{j \in J_0} w_j$ be their total weight. We discard any set not used in the course of the dynamics.

With each possible location of a chip at some height i (from bottom) in some stack j, we assign a *position* of *value*[11] w_j/i. Thus in any state, a chip's cost equals the value of its current position. The plan is to bound the cost of the m most expensive chips by bounding costs of expensive positions and moves among them.

[10]Note however that the game may still be non-symmetric, i.e. different players may have different available sets.

[11]To avoid confusion, we talk about the weight of a set, the cost of a chip and the value of a position

It is easy to see that any set has weight at most $W_0(1+\epsilon)^{2(m-1)}$ (clearly the case for sets in J_0). Indeed, whenever a player moves to a previously unoccupied set j' from a set j, the weight of j' is at most $(1+\epsilon)^2$ times the weight of j; one can trace back each set to an initial set using at most $m-1$ steps (there are m sets in all). We also claim that at most $mi(1+\epsilon)^{2m}$positions have value at least $\frac{W_0}{i}, \forall i$: indeed positions of height $i(1+\epsilon)^{2m}$ or more on any set have value less than $\frac{W_0}{i}$ since any set has weight at most $W_0(1+\epsilon)^{2(m-1)}$.

Fix a constant $C > (1+\epsilon)^{2m}$ (recall $\epsilon = O(\frac{1}{m})$). Note that any chip on a position of value less than $\frac{W_0}{m}$ in S_0 never achieves a cost greater than $\frac{W_0}{m}(1+\epsilon)^{2Cm^2}$. Indeed, by the reasoning above for $i = m$, there are at most $m \cdot m \cdot (1+\epsilon)^{2m} \leq Cm^2$ positions of greater value. Thus the chip's cost never exceeds $\frac{W_0}{m}(1+\epsilon)^{2Cm^2}$as it can increase at most Cm^2 times (by an $(1+\epsilon)^2$ factor).

We upper bound the total cost of the *final* m most expensive chips, as it is no less than the final social cost: for a set, its weight equals the cost of its most expensive chip. We reason based on chips' initial costs. Namely, we claim $h(i) \leq \frac{W_0}{i-1} \cdot (1+\epsilon)^{2Cm^2}, \forall i \in [m]$, where $h(i)$ denotes the cost of ith most expensive chip in the final configuration. If this chip's initial cost is less than $\frac{W_0}{m}$ then the bound follows from the claim above. Now consider all chips with an initial cost at least $\frac{W_0}{m}$. As argued above, at most Cm^2 positions have value $\frac{W_0}{m}$ or more, and any of these chips increased in cost by at most $(1+\epsilon)^{2Cm^2}$. A simple counting argument shows that for any i, there are at most i chips of initial cost at least $\frac{W_0}{i}$ and thus $h(i) \leq \frac{W_0}{i-1} \cdot (1+\epsilon)^{2Cm^2}, \forall i$.

To show this, we claim that for any k, there are at most k chips of initial cost at least $\frac{W_0}{k}$. Let J_0, be the set of initially used resources. For each $j \in J_0$, let r_j be set j's fraction of the initial weight (i.e. $w_j = r_j W_0$), and let p_j be the number of initial positions with value greater than $\frac{W_0}{k}$ in set j. We have $\frac{w_j}{p_j} = \frac{r_j W_0}{p_j} \geq \frac{W_0}{k}$, implying $p_j \leq k r_j$. Counting the number of initial positions with sufficient value yields $\sum_j p_j \leq \sum_j k r_j = k \sum_j r_j = k$ since $\sum_j r_j = 1$.

As the i^{th} most expensive chip has cost at most $\frac{W_0}{i-1}(1+\epsilon)^{2Cm^2}$(at most $i-1$ chips have higher final cost),

$$\sum_{i=1}^{m} h(i) = h(1) + \sum_{i=2}^{m} h(i) \leq h(1) + \sum_{i=2}^{m} \tfrac{W_0}{i-1}(1+\epsilon)^{2Cm^2}$$
$$= O(W_0(1+\epsilon)^{2m} + W_0(1+\epsilon)^{2Cm^2} \log m) = W_0 \cdot (1+\epsilon)^{O(m^2)} O(\log m)$$

As desired, $PoU_{IR}(\epsilon, \textit{set-covering}) = (1+\epsilon)^{O(m^2)} O(\log m)$ as the final social cost is at most $\sum_{i=1}^{m} h(i)$. □

The previously known bound [12] is $PoU_{IR}(\epsilon) = O((1+\epsilon)^{2mn} \log n)$. Unlike our bound, it depends on n (exponentially) and it does not guarantee a small $PoU_{IR}(\epsilon)$ for $\epsilon = \Theta(\frac{1}{m^2})$ and $m = o(n)$. This bound uses chips in a less sophisticated way, noting that any chip can increase its cost (by $(1+\epsilon)^2$) at most mn times.

2.5.1.1 Best-Response in Matroid Congestion Games

In this section we extend this bound to best-response dynamics in matroid congestion games. These games are important in that they precisely characterize congestion games for which arbitrary best-response dynamics (without uncertainty) converge to a Nash equilibrium in polynomial time [1]. We show that their BR dynamics can be simulated by IR dynamics in *generalized set-covering* games, in which resources may have arbitrary latencies, but each player must use exactly one resource.

We first define a *congestion game* [66], comprised of a set of resources $1 \ldots m$ with $\Sigma_i \subseteq 2^{1 \ldots m}$ specifying the strategies (i.e. resource subsets) that player i may use. Each resource j has a delay function $c_j : \mathbb{N} \to \mathbb{R}$ (not necessarily increasing) that depends only on the number of players using j. A player's cost is defined as the sum of the delays on all resources he uses. A congestion game has a canonical potential defined as $\Phi(S) = \sum_{j=1}^{m} \Phi^j(S)$, for any state S where $\Phi^j(S) = \sum_{i=1}^{n_j(S)} c_j(i)$ denotes the aggregated cost if the $n_j(S)$ current users of j joined j sequentially: $c_j(1)$ for the first player, $c_j(2)$ for the second player etc. Set-covering games are congestion games; the potential in Eq. (2) is a special case of the one defined here.

A *matroid* is a combinatorial structure generalizing, among others, sets of linearly independent rows of a matrix and cycle-free sets of edges in a graph. Formally, a *matroid* is a tuple $M := (1 \ldots m, \mathcal{I})$ where $\mathcal{I} \subseteq 2^{1 \ldots m}$ is a family of *independent* subsets of $1 \ldots m$, such that $\mathcal{I}$ is hereditary, i.e. if $I \in \mathcal{I}$ and $J \subseteq I$ then $J \in \mathcal{I}$, and $\mathcal{I}$ satisfies the exchange property, i.e. if $I, J \in \mathcal{I}$ and $|I| > |J|$, then $\exists i \in I$ such that $J \cup \{i\} \in \mathcal{I}$. All maximal sets of $\mathcal{I}$ (*bases* of M) have the same size (by the exchange property), called M's *rank* $rk(M)$. In a *matroid congestion game*, a player i's strategy set Σ_i is the set of bases of a matroid M_i on the resources.

There is a natural representation [12] of BR dynamics in a matroid congestion game as an instance of IR dynamics in a generalized set-covering game over the same m resources. Each player i controls $rk(M_i)$ markers, one for each resource she currently uses. A move from one basis to another corresponds to a player moving (some of) her markers from her current resources to her new resources. Thus a PoU_{IR} upper bound on generalized set-covering games automatically implies a PoU_{BR} upper bound for matroid games.

Our next result is an immediate extension of Theorem 4 to a generalization of set-covering games, Fair Cost Sharing Matroid Games (FCSMG), where each resource has a base cost, and the cost of a resource is split evenly by all who use it. Theorem 4's upper bound is, importantly, independent of the number of players and thus not affected by the increase from n players to $\sum_{i=1}^{n} rk(M_i)$ marker players.

Theorem 5. *$PoU_{BR}(\epsilon, FCSMG) = (1+\epsilon)^{O(m^2)} O(\log m)$ for $\epsilon = O(\frac{1}{m})$.*

This is a corollary of Theorem 4.

Proof. Note that we can view the dynamics as an improved-response set-covering game, with m sets and $\sum_{p=1}^{n} rk(M_p)$ players. However, our bound does not depend on the number of players, so this follows immediately. $\square$

We now generalize this upper bound to a much broader class, that of matroid congestion games with decreasing cost functions (MCG$_D$). We consider an alternative cost function for sets as an intermediary in our proof: $cost'(S) = \sum_{j:n_j(S) \geq 1} c_j(1)$, i.e. the sum of the "base cost" $c_j(1)$ of each resource j used.

We define $\text{GAP}' = \max_j \frac{\sum_{i=1}^{n} c_j(i)}{c_j(1)}$ as the gap between potential and the new social cost function. For any resource j, this is the ratio of its contribution to the potential to its contribution to the social cost. By considering all resources we get $cost'(S) \leq \Phi(S) \leq \text{GAP}' \cdot cost'(S)$. Since for decreasing delay functions, the standard social cost is less than the potential we get $cost(S) \leq \Phi(S) \leq \text{GAP}' \cdot cost'(S)$.

Theorem 6. *For $\epsilon = O(\frac{1}{m})$, $PoU_{BR}(\epsilon, MCG_D) = (1+\epsilon)^{O(m^2)} O((\text{GAP}')^2 \log m)$.*

Proof. We first note that the increase in potential of any marker is limited by the number of positions of greater potential. This is done the same as in Theorem 4.

We then carefully count the positions of large potential and show that there can only be a few large markers. This will bound the overall increase of base cost. Finally, we bound the difference between the social cost and base cost.

Let W_0 be the social cost of the initial state, an upper bound on the base cost of the initial configuration.

As delay functions are decreasing, we can note that when a player moves to an unoccupied resource j' from some previously occupied resource j that the base cost of j' is at most $(1+\epsilon)^2$ greater than the base cost of j. Thus the base cost for any set does not exceed $W_0(1+\epsilon)^{2m}$. The total potential (for all sets) is then bounded by $m\text{GAP}' \cdot W_0(1+\epsilon)^{2m}$.

The total potential is also the sum of all positions' values. We note that there are at most $mk(1+\epsilon)^{2m}$ positions of value exceeding $\text{GAP}' \cdot W_0/k$. In particular, there are at most $m^2(1+\epsilon)^{2m}$ positions of value greater than $\text{GAP}' \cdot W_0/m$. As $\epsilon = O(\frac{1}{m})$, we can define the constant $C = (1+\epsilon)^{2m}$.

Since we are bounding the base cost it suffices to bound the value of the m heaviest markers. Note that if there are at most x positions of value greater than v, then the value of any marker with initial value less than v is bounded by $v(1+\epsilon)^{2x}$. Thus all marker with initial value less than $\text{GAP}' \cdot W_0/m$ never achieve a value exceeding $\text{GAP}' \cdot W_0(1+\epsilon)^{2m^2C}/m$. Furthermore, any markers with initial cost greater than this never increase by a factor of more than $(1+\epsilon)^{2m^2C}$.

Thus to bound the m heaviest markers at some point in the dynamics, it suffices to bound the cost of the m heaviest markers initially. We simply observe that the kth most expensive chips initially cannot weigh more than $(1/k)\text{GAP}' \cdot W_0$ individually. Summing, we find that the total cost of these markers amounts to $\textsc{gap}' \cdot W_0 O(\log m)$, as desired. Thus we deduce that the base cost of any configuration is bounded by $(1+\epsilon)^{2m^2C}\textsc{gap}' \cdot W_0 O(\log m)$. We use this to bound the social cost. Since the cost function is decreasing, the potential of a set is greater than the actual social cost of the players on the set. As $\textsc{gap}'$ bounds the difference between the base cost and the potential, the social cost is $(1+\epsilon)^{2Cm^2}W_0(\textsc{gap}')^2O(\log m)$. Thus the price of uncertainty can be bounded by $(1+\epsilon)^{2Cm^2}(\textsc{gap}')^2O(\log m)$, as desired. $\square$

Let us now investigate $\textsc{gap}'$, for some simple decreasing cost functions. Consider a matroid congestion game, with delay functions of the form $c_j(x) = \frac{k_j}{x^\alpha}$, with $\alpha \in (0,1), k_j \in \mathbb{R}, \forall j$. We have that $\Phi^j(S)\sum_{l=1}^{n}\frac{k_j}{l^\alpha} \approx k_j\int_1^n x^{-\alpha}dx = k_j\frac{1}{1-\alpha}n^{1-\alpha}$ i.e. within a constant factor (namely $\frac{1}{1-\alpha}$) of the social cost. Thus $\textsc{gap}' = \Theta(n^{1-\alpha})$. This implies a bound of $PoU = (1+\epsilon)^{O(m^2)}O(n^{2-2\alpha}\log m)$ these games. The existing upper bound was exponential in n (rather than polynomial, as in our bound) for many values of ϵ.

2.5.2 Lower Bound for Improved-Response

Our upper bound showed that $PoU_{IR}(\epsilon)$ is logarithmic for $\epsilon = O(\frac{1}{m^2})$. A basic example (one player hopping along sets of cost $1, (1+\epsilon)^2, \dots, (1+\epsilon)^{2(m-1)}$), applicable to many classes of games, yields the lower bound $(1+\epsilon)^{2(m-1)} \leq PoU_{IR}(\epsilon, \textit{set-covering})$. In fact, this immediate lower bound is the best known on $PoU_{IR}(\epsilon)$. For $\epsilon = \omega(\frac{1}{m})$, we get that $PoU_{IR}(\epsilon)$ is large. An intriguing question is what happens in the range $[\omega(\frac{1}{m^2}), \Theta(\frac{1}{m})]$, in particular for natural uncertainty magnitudes such as $\epsilon = \Theta(\frac{1}{m})$ or $\epsilon = \Theta(\frac{1}{n})$.

In this section we show that for $\epsilon = \Theta(\frac{1}{\min(m,n)})$, PoU can be as high as polylogarithmic. We obtain this by a non-trivial construction that repeatedly uses the snowball effect to locally increase one chip's cost, without other changes to the state. Our main gadget is a *pump*, treated for now as a black box and described fully in Section 2.5.2.1. A pump increases a chip's cost by $\alpha = \log n'$, where $n' = \min(m,n)$. We use p pumps to increase each chip's cost by a $\Omega(\log^p n')$ factor. As pumps are "small", the total cost increase is $\Omega(\log^p n')$.

Theorem 7. *$PoU_{IR}(\epsilon, \textit{set-covering}) = \Omega(\log^p \min(m,n))$, for $\epsilon = \Theta(\frac{1}{\min(m,n)})$ and constant $p > 0$.*

We first define a key component of our construction. An (α, W)-pump uses $O(\frac{1}{\epsilon})$ sets and $O(2^\alpha)$ players to increase, one by one, an arbitrary number of chip costs by an α factor from W/α to W. For ease of exposition, we assume $m = \Theta(n)$ and we only treat $p = 2$, i.e. how to achieve $PoU_{IR}(\frac{1}{n}) = \Omega(\log^2 n)$. For general p, we use p pump gadgets instead of two.

Definition 3 (Pump). *An (α, W)-pump P is an instance of a set-covering game specified as follows:*

- *The number m_P of sets used is $O(\frac{1}{\epsilon})$. For our choice of ϵ, $m_P = O(n)$. The total weight W_P of all sets in P that are initially used is in $(2^\alpha W, e2^\alpha W)$. The number of players used is $n_P = 2^{\alpha+1} - 2$.*
- *Within $O(n^3)$ moves of IR dynamics contained within the pump, and with a final state identical to its initial state, a pump can consume any chip of cost at least W/α to produce a chip of cost W.*

Proof. Let $N := \alpha^2 2^\alpha$. The number of players will be $n := N + n_{P_1} + n_{P_2}$. Thus $\alpha = \Theta(\log n)$. Note that each player can use each set.

We use two pumps, an $(\alpha, 1/\alpha)$ pump P_1, and a $(\alpha, 1)$ pump P_2. Aside from the pumps, we have Type-I, Type-II and Type-III sets, each with a weight of $1/\alpha^2$, $1/\alpha$ and 1 respectively. At any state of the dynamics, each such set will be used by no player or exactly one player. In the latter case, we call the set *occupied.* We have N Type-I sets, 1 Type-II set, and N Type-III sets, i.e. $m := 2N + 1 + m_{P_1} + m_{P_2} = \Theta(n)$ sets in all.

Let cfg(i, j, k) refer to the configuration with i Type-I sets occupied, j Type-II sets occupied, and k Type-III sets occupied. We shall use $2N + 1$ intermediate states, denoted $state_i$. Our initial state is $state_0 = \text{cfg}(N, 0, 0)$, and our final configuration will be $state_{2N} = \text{cfg}(0, 0, N)$. In general, $state_{2i} = \text{cfg}(N - i, 0, i)$ and $state_{2i+1} = \text{cfg}(N - i - 1, 1, i)$. Thus we want to move each player on a Type-I set (initially) to a corresponding Type-III set, an α^2 *increase* in cost. To this purpose, we will pass each such player through the first pump and move it on the Type-II set. This achieves the transition from $state_{2i}$ to $state_{2i+1}$. Since the player's cost is increased by an α factor (from $\frac{1}{\alpha^2}$ to $\frac{1}{\alpha}$), we can pass it through the second pump and then move it on the Type-III set. This achieves the transition from $state_{2i+1}$ to $state_{2i+2}$.

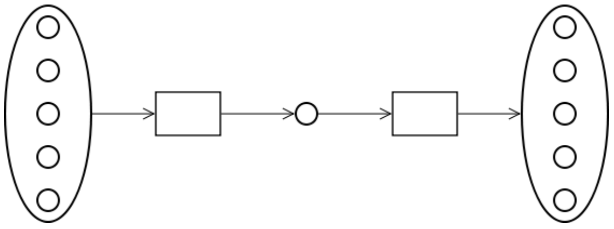


Figure 4: This shows the structure of the construction, with pumps used as black boxes. Players travel from sets of Type-I, to the Type-II set, and then to a set of Type-III.

The social cost of our initial configuration is $W_0 = N \cdot \frac{1}{\alpha^2} + W_{P_1} + W_{P_2} \leq 2^\alpha + e \cdot \frac{1}{\alpha} 2^\alpha + e \cdot 2^\alpha \leq 7 \cdot 2^\alpha$. The final social cost (excluding the pumps) is at least $N = \alpha^2 2^\alpha$. Thus $PoU = \alpha^2$, and $\alpha = \Theta(\log n)$.

Finally, we note the number of moves in the dynamics. Note that there are $2N$ uses of the pumps. As we shall show, each use of the pump uses fewer than n^3 moves. Thus in all we use at most $O(n^4)$ moves. $\square$

We now detail an $(\alpha, \cdot)$ pump (that we call an α-level pump). Fig. 5 depicts a pump with $\alpha = 4$.

2.5.2.1 Pump Gadget

We first outline the mechanism of the pump and then define it more formally.

- A pump has chips of costs $w, \frac{w}{2}, \ldots, \frac{w}{\alpha}$ (at heights $1 \ldots \alpha$) on sets of weights $w = W, W(1+\epsilon), \ldots, W(1+\epsilon)^{\frac{1}{\epsilon}}$
- By repeated advancements (see Algorithm 1), the pump doubles the cost of each chip.
- One chip at each level is promoted to next level, and all other chips return to their starting configuration.

We make several conventions for a more intuitive exposition. We may refer to sets as stacks; accordingly, level k will be the collection of chips all at height k in their respective stacks. A chip in level k will always maintain its level, except possibly at the end of the pump. When talking about chips on a given level, we say that any set with no players on the relevant level is *unblocked.* We assume that α and $\frac{1}{\epsilon}$ are integers.

A pump has $\frac{1}{\epsilon}+1$ sets, and α special sets, i.e. $m_P = \frac{1}{\epsilon}+\alpha+1$. We shall number the main sets $s_0, s_1, \ldots, s_{\frac{1}{\epsilon}}$, and the special sets $t_1, \ldots, t_\alpha$. Set s_i will have weight $W_i = W(1+\epsilon)^i$. Each set t_i will have weight $2W/i$. We refer to all main sets simply as sets. Fig. 5 only depicts the main sets (as gray blocks). Special sets will only be used during the reset phase. We get $2^\alpha W \leq W_P \leq e2^\alpha W$ by noting that the weight of each of the 2^α occupied sets is in $[W, eW]$. There are $n_P = \sum_{j=1}^{\alpha} 2^j = 2^{\alpha+1} - 2$ players in the pump. Each player will be allowed to use each set of the pump, as will any players who move into the pump.

In the pump's initial state (see Fig. 5a), the first 2^α sets will be occupied. The first $2^{\alpha-j}$ sets will each be occupied by (at least) $j+1$ chips, except for the first set s_0, which will "save" one position for the input chip and thus will only be occupied by $\alpha - 1$ chips. To activate the pump, first a chip of cost at least W_P/α moves on to the top of stack 0. This is followed by $\frac{1}{\epsilon}$ "advancement" phases, and then a reset phase.

Each advancement phase moves the entire collection of players one set forward on the pump. An advancement phase is depicted in Fig. 5a to Fig. 5k. Note that, since the sets are each a $(1+\epsilon)$ factor apart, any chip can be induced to move to either of the next *two* sets (on the same level). The chip may also freely move to any set (on the same level) which is lower.

The players in the advancement phase are moved in accordance with Algorithm 1. Each advancement phase for level i starts with an advancement for level $i+1$ (seen in Fig. 5b, 5c and 5f for $i = 3, 2, 1$). Then all level $i+1$ chips are positioned on even sets (seen in Fig. 5b, 5d and 5i for $i = 3, 2, 1$). This allows level i chips on odd sets, one by one, to advance two sets, effectively moving one set from the start of level i to the end of level i (seen in Fig. 5a to 5b, 5b to 5c, 5d to 5f and 5i to 5j for $i = 4, 3, 2, 1$).

Note that after $\frac{1}{\epsilon} - 2^\alpha$ phases the bottom level has no more space to move forward. Henceforth we apply an $\alpha - 1$ advancement. In general, if only L levels still have empty space then we only advance L levels.

The reset phase (not shown in Fig. 5) is simpler. At this stage there are α chips on the last set $s_{\frac{1}{\epsilon}}$. We move the chip at height j in this last stack, in decreasing order of j, to the special set t_j. This is a valid IR move since the chip's cost $(1+\epsilon)^{\frac{1}{\epsilon}} W/j$ is at least the weight $2W/j$ of t_j. Since $2W/j \geq W/(j-1)$ for $2 \leq j \leq \alpha$, we may move the chip in t_j to the $(j-1)$st level of set 0. Subsequently, we repeatedly move the unblocked chip of lowest level to the first unoccupied set in that level. When this is finished, we move the chip on t_1 out of the pump. The pump has returned to its initial state via moves inside the pump only. The pump consumed a chip of cost at least W/α, and produced a chip of cost W, as desired.

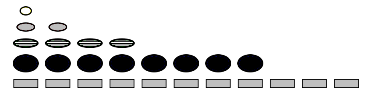
Finally, note that one use of a pump employs $O(n^3)$ moves: for each of the $\Theta(\frac{1}{\epsilon})$ advancement phases, each of the $2^{\alpha+1}$ chips moves at most 2^α times. The reset phase has fewer moves than an advancement one.

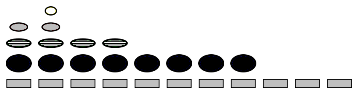
Algorithm 1 describes a level advancement phase for a pump.

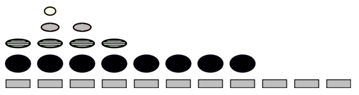
Algorithm 1 k Level Advancement.

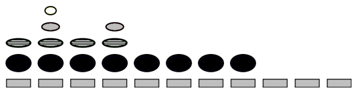
```
 1: for $i = \alpha, \alpha-1, \ldots, k+1$ do
Require: All level $i+1$ chips to be on even sets up to $2^i$ (i.e. all chips on odd sets in
      level $i$ to be unblocked).
 2:   while There are still two level $i$ chips adjacent do
 3:     Take the level $i$ chip on the largest odd set, $j$.
 4:     if set $j+1$ is unoccupied on level $i$ then
 5:       Move the chip to set $j+1$
 6:     else
 7:       Move the chip to set $j+2$
 8:     end if
 9:   end while
Ensure: Each chip on level $i$ is on even sets up to $2^{i+1}$.
10: end for
11: Iterating through odd sets in decreasing order, move each level $k$ chip forward by 2.
12: while There are two non-adjacent chips in a level do
13:   Of chips that are both unblocked and not in a single contiguous level, choose the chip
      with the lowest level.
14:   Move this chip to the lowest set it can occupy.
15: end while
```

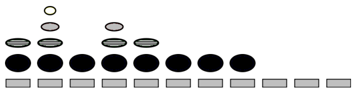
As we show below, our pump gadget can be tweaked to also provide a polylogarithmic lower bound on PoU for generalized set-covering games with *increasing* delay functions, as long as they have bounded jumps, i.e. if an additional user of a resource cannot increase its cost by more than a constant factor. This well-studied class is much broader than

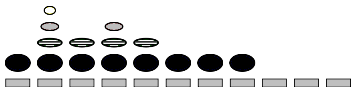

(a) The initial configuration for a pump. Each level has twice the number of chips of the level that rests on it.

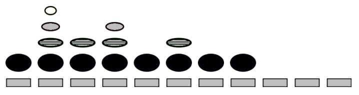

(b) We begin with a level 4 step forward.

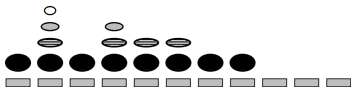

(c) After a level 3 chip hops forward.

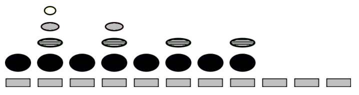

(d) Having arranged the light gray chips on alternating sets, we can now hop the dark grays forward.

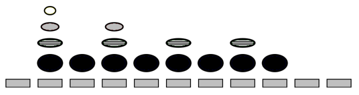

(e) First the rightmost dark gray makes room ...

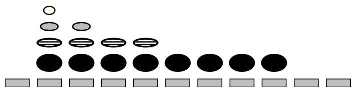

(f) ... and then the leftmost dark gray completes the level 2 step forward.


(g) It now remains to position the dark grays on alternating sets so as to allow the black sets to advance.


(h)


(i) We finally arrange the dark grays on alternating sets.


(j) Finally with all black odd sets unblocked, each black odd set, starting from the right, hops forward two spaces.


(k) All players can return to the initial position — but shifted over a single space — and the process repeats.

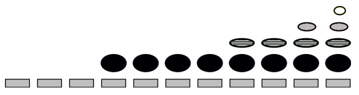

(l) In this final position, the starting cost of any chip has roughly doubled, a greater increase than the ratio between levels. We then promote one node of each level to the lower level, and reset the pump.

Figure 5: A 4-level ($\alpha = 4$) pump. Each gray block is a set, and each set has a weight that is precisely a factor of $(1+\epsilon)$ greater than its predecessor. Note that any chip can move up to two spaces forward (at the same level) and hop as far back as desired (on the same level). This sequence shows the advancement procedure, listed in Algorithm 1, by which all levels move forward.

set-covering games (that have decreasing delay functions).

Let $c_e(x)$ be the cost function for edge e with x players on it. We shall consider edges with $c_e(x) = k_e d(x)$, where k_e is an edge-dependent multiplier, and $d(x)$ is a base delay function that is identical across all edges.

First, let us augment the definition of a pump with a length parameter ℓ, such that the total number of sets in the pump is $\alpha + \ell$ instead of $\frac{1}{\epsilon}$. The pump has, as before, $\Theta(2^\alpha)$ players. The pump can consume any chip of weight $W/d(\alpha)$ to produce a chip of weight W.

The pump is functionally the same as before: players stay on the same level, and may advance past each other as before. In this pump, the levels of highest value are on the top, rather than the bottom, since delay functions are increasing (recall for set-covering we called a given (set, level) pair a position). As before, we number the levels in increasing order of value: level k contains $2^{\alpha+1-k}$ players. As before, the pump is valid for any weight W.

For a pump to operate, a chip that travels the length of the pump must increase its cost enough to enter the next level. Then we only require that

$$\frac{c_e(x+1)}{c_e(x)} \leq (1+\epsilon)^\ell \forall x = 1 \dots n-1 \tag{3}$$

The reader may recognize Eq. (3) as the γ-*bounded jump* condition (see for example Chien and Sinclair [28]) with $\gamma = (1+\epsilon)^\ell$. Note that $\frac{c_e(x+1)}{c_e(x)} = O(1)$ for polynomial ($c_e(x) = k_e x^{r-1}$) and even for exponential ($c_e(x) = k_e r^x$) delay functions (where $r > 1$), that we will revisit shortly.

Theorem 8. *For generalized set-covering games with increasing delay functions with γ-bounded jumps (where $\gamma \in (1, \infty)$) $PoU_{IR} = \Omega(\log^p n)$ for any fixed constant p, for $\epsilon = \Theta(1/n)$ and $m = \Theta(n)$.*

Proof. We will use a similar, but simpler, construction compared to the one in Theorem 7 for set covering games. We have one pump (instead of two), n sets of weight W and n sets, initially occupied, of weight $W/d(\alpha)$. We bound the price of uncertainty as follows.

Initially, we have $\alpha 2^\alpha$ sets of weight $W/d(\alpha)$ with one player each, and a pump with 2^α players of weight at most W. In total, the initial weight is at most $2^\alpha W(\alpha/d(\alpha) + 1)$.

Our final weight is at least $\alpha 2^{\alpha} W$. We have constructed it so that $\alpha \approx \log n$.

$$PoU \geq \frac{\alpha}{\alpha/d(\alpha)+1} = \frac{\alpha d(\alpha)}{\alpha + d(\alpha)} \geq \frac{1}{2}\min(\alpha, d(\alpha)).$$

□

In particular, for polynomial delay functions $d(n) = c \cdot n^{\delta}$ with $\delta \in \mathbb{N}$,

$$\tfrac{d(n+1)}{d(n)} = \tfrac{c(n+1)^{\delta}}{cn^{\delta}} = \left(\tfrac{n+1}{n}\right)^{\delta} \leq 2^{\delta}.$$

Thus the pump construction is valid for $\ell \geq \frac{1}{\epsilon}\delta \ln 2 = \Theta(1/\epsilon)$, as desired.

Another interesting case is that of exponential delay functions [75]. If $d(n) = r^n$, then it suffices for the pump to have length $\ell = \frac{1}{\epsilon}\ln r = \Theta(1/\epsilon)$. This is interesting because our PoU lower bound is slightly better in this case. Computing the initial weight of the pump we find that it is of order αW rather than our naïve bound of $2^{\alpha} W$.

2.5.3 Lower Bound for Best-Response

A significant increase in costs is possible for a large range of ϵ even if players follow best-response dynamics.

Theorem 9. *$PoU_{BR}(\epsilon, \text{set-covering}) = \Omega(\epsilon n^{1/3}/\log n)$, for any $\epsilon = \Omega(n^{-1/3})$. This holds for any arbitrary ordering of the dynamics, i.e. no matter which player is given the opportunity to update at any time step.*

Proof. The construction is quite similar in nature to the previous construction of Balcan et al. [12] which achieves a lower bound of $\Omega(\epsilon\sqrt{n}/\log n)$. While our lower bound has a weaker guarantee, it will preserve its guarantee even under arbitrary orderings, unlike the aforementioned construction. Set $N := n^{2/3}$. The increase in social cost in this construction comes from N players sitting on $\sqrt{N}$ sets (of cost N) being moved to individual sets of cost N. We shall introduce other players and sets to facilitate this, taking care that their initial cost is not too high.

Consider N players of Type I, indexed by pairs (i, j) for $1 \leq i, j \leq \sqrt{N}$. Type-I player (i, j) has 3 sets to choose from: a set s_i^*, a set s_j^i and a personal set $s_{(i,j)}$, all of cost N.

Initially, all Type-I players begin on the sets s^*_i, for a total cost of $N\sqrt{N}$. We shall call the sets s^i_j the *active* sets, the sets s^*_i the *base* sets, and the sets $s_{(i,j)}$ the *final* sets.

Next there are N players of Type II, indexed by pairs (j,k), with $1 \le j \le \sqrt{N}$, and $2 \le k \le \sqrt{N}$. A Type-II player (j,k) may choose any of the sets s^i_j, and has an additional set $f_{(j,k)}$. The cost of $f_{(j,k)}$ is $\frac{1}{\epsilon}\frac{N}{k}$. All Type-II players begin on the sets $f_{(j,k)}$. In total, the initial cost of these sets is $(\frac{1}{\epsilon}N\sqrt{N})\log N$. We refer to the sets $f_{(j,k)}$ as *private* sets.

Additionally on each set $f_{(j,k)}$ there are $1/\epsilon$ 'helper' players of Type III. These players are indexed by triples, where the first pair indicates the Type-II player whose set they are associated with. Each Type-III player (j,k,l) may use only two sets, the private set $f_{(j,k)}$ and a set $f_{(j,k,l)}$, with cost $(\frac{1}{\epsilon}\frac{N}{k})/l$. (For players with $l < \frac{1}{2\epsilon}$, we shall only allow them to use the private set, thus ensuring that each private set has at least $\frac{1}{2\epsilon}$ players on it at all times.) We shall refer to the sets $f_{(j,k,l)}$ as *storage* sets. Type-III players all begin on the sets $f_{(j,k)}$.

As in the previous construction the Type-III players drive the Type-II players to the active sets, which then lure the Type-I players (from one base set) to the active sets. We call this the loading phase. Once the base set is empty, the Type-II and Type-III players return to their starting set, and the Type-I players move to their final set. We call this the unloading phase. Then the process repeats with a different base set.

We first examine the loading phase. The adversary first queries all Type-III players in reverse lexicographical order, then all Type-II players in lexicographical order, and then finally all Type-I players in any order.

Consider a Type-III player (j,k,l). They have a storage set of cost $s := (\frac{1}{\epsilon}\frac{N}{k})/l$. The apparent cost of the private set is $p := (\frac{1}{\epsilon}\frac{N}{k})/(l+1)$. Since $l \ge \frac{1}{2\epsilon}$, we see that $s < p(1+\epsilon)^2$, so this perturbation will compel the Type-III player to move to its storage set.

Next the Type-II players move. Consider a Type-II player (j,k). Since $1/(2\epsilon)$ players have moved off of every private set, the apparent cost is now $2N/k \ge N/(k-1)$. Thus querying the players in lexicographical order will lead to each Type-II player moving to the active sets. Note that the players $(j,2)$ have apparent cost N and able to choose from all the active sets. The adversary will ensure (via a small perturbation) that they choose active sets s^i_j with i corresponding to the base set being moved.

Finally the Type-I players move. The players on base set i each see an available choice with apparent cost $\sqrt{N}$ — namely the active set, and this is their best response. Type-I players from other base sets are either alone on a final set and see empty sets of cost N, or stay on a set of cost $\sqrt{N}$.

During the unloading phase, the adversary moves first the Type-III players onto the private sets in lexicographical order. Then he moves the Type-II players onto the private sets in reverse lexicographical order, and finally the Type-I players onto the final sets in any order.

The adversary first queries the Type-III players in lexicographical order. They are indifferent between the private sets and their storage sets, so an arbitrarily small perturbation causes them to return to the Type-II players sets. Once all the Type-III players have returned, the Type-II players can return in decreasing order, as they will be indifferent between their choices.

Finally, when all Type-II players have returned to their private sets, the Type-I players move. They only players willing to move are on active sets, and they have a choice of three sets. Each one of these sets has no other players and has a cost of N, so an arbitrarily small perturbation will cause them to choose the final set.

Thus each loading phase and unloading phase transfers players from one base set to the final sets. After $\sqrt{N}$ repetitions, all Type-I players have moved to the final sets, and all other players are on the private sets.

Finally, we compute the PoU. First we note that the initial cost of the sets is $\Theta(N\sqrt{N} + 1/\epsilon N\sqrt{N}\log N)$. The final cost is at least N^2 from Type-I players alone. Thus the increase is $\Omega(\epsilon\sqrt{N}/\log N)$. We note that this construction uses $\Theta(N/\epsilon) = O(n)$ players and $m = \Theta(N/\epsilon) = O(n)$ sets. Since $N = n^{2/3}$, we can substitute to find $PoU_{BR} = \Omega(\epsilon n^{1/3}/\log n)$ as desired. □

In particular, we observe that $PoU_{BR}(\epsilon, SCG) = \Omega(n^{1/6}/\log n)$ for $\epsilon = n^{-1/6}$ if the adversary can choose which player to update at any given step. Our next theorem will show that this result remains valid for any ordering of the dynamics, even one that the adversary cannot influence.

Theorem 10. *The construction in Theorem 9 is valid regardless of the ordering of updates, provided that the ordering guarantees, at every point in time, that each player will be given a future chance to update.*

Proof. We shall prove this by showing that each loading and unloading phase completes as planned regardless of the ordering. To show this, note that once all Type-II and Type-III, and Type-I players have moved once, the phase is over. (Note that we restrict our attention to 'active' Type-I players, or those whose base set is being unloaded.) Thus it suffices to show that none of them ever move twice in a phase.

This is simplified by the restricted set of choices of the players. No player can use both storage sets and active sets, so the interactions between sets are limited to players moving between storage sets and private sets, or moving between private sets and active sets.

We shall show this by a simple inductive argument over each loading and unloading stage, by showing that, in each stage, players move only one direction relative to the sets, either all joining or all leaving.

During the loading stage, the adversary can ensure that:

1. The apparent cost of the base set only increases.

2. The apparent cost of each active set only decreases.

3. The apparent cost of each private set only increases.

We show that no player moves off of an active set, no player moves onto a private set, and that no player moves onto a base set.

First consider a player moving onto its base set. The only players who can consider this are Type-I players who have already moved to an active set. In that case, however, they are on an active set whose apparent cost has only decreased since they chose, while the cost of the base set has only increased. The same perturbations which led them to choose the active set before will cause them to make the same choice again.

Now consider a player who would move off of an active set. This player cannot be Type-I, so they must be Type-II. Again, we note that they moved to the active set during this phase, and its cost has only decreased while the cost of their base set has increased. The perturbations which led them to choose the active set before will cause them to choose it again.

Finally we note that the cost of a private set only increases during this phase. We know that the adversary can ensure that no Type-II players join the private set, so it suffices to show that no Type-III players join it. But at the beginning of the phase they all started on the private set. By using the same perturbations that caused them to choose the storage set, any Type-III player considering moving from a storage set to a private set can be influenced not to.

Once all Type-II players, Type-III players and active Type-I players have moved once, the phase is complete and the unloading phase begins.

During the unloading stage, the adversary ensures that:

1. The apparent cost of each active set only increases.

2. The apparent cost of each private set only decreases.

To maintain this invariant it suffices to show that no player will move from a private set to a storage set, and that no player will move from a private set to an active set.

First we show that the cost of a private set does not increase. Consider a player on a private set who considers moving to an active set. This player must be of Type-II, but note that earlier in this unloading stage they moved from an active set to the private set. The private set has only decreased in apparent cost, and the active set has only increased in apparent cost. Therefore the perturbations that made this player choose the private set before will make that player choose the same set again.

Consider a player on a private set who considers moving to a storage set. This player is of Type-III, and must have an $l \geq 1/2\epsilon$. They began the unloading phase on a storage set, but chose to move to the private set. Since the apparent cost of the private set has only decreased, (and the cost of the storage set is unchanged) the adversary can compel them to make the same choice as before.

No Type-I player will want to move unless they are alone on an active set. In this case, the adversary can compel them to choose the final set via an arbitrarily small perturbation.

We have thus demonstrated that with any ordering of updates, the loading and unloading phases may proceed as before. Thus the construction still holds, for arbitrary ordering rules. □

Let us take a moment to outline some of the key differences between this construction and the previous lower bound [12]. The structure is very similar. Because they allow the adversary to control the order of updates only needs $O(N)$ players of Type-II. This smaller set of Type-II players lures each Type-I player off a base set sequentially, and doesn't let them update until all Type-I players have been removed from the base set. Thus the increase to $O(N^2)$ comes from a lower bound of order $\Omega(\frac{1}{\epsilon}N\log N)$ rather than our $\Omega(\frac{1}{\epsilon}N^2\log N)$. Additionally, we have $O(\frac{1}{\epsilon}N^2)$ players rather than only $O(\frac{1}{\epsilon}N + N^2)$, so our constraints on ϵ must be traded off directly against N. Matching the previous lower bound of $PoU \geq (\frac{\epsilon\sqrt{n}}{\log n})$ in the arbitrary ordering model would require new insight into the structure of this game.

2.6 Best-response in Random Order in (λ, μ)-smooth Games

Throughout most of this chapter we have studied *best-response* dynamics in which the adversary controls the order of player updates. We have shown this is not necessary for lower bounds in other contexts, but we will now investigate a game in which random order helps keep the Price of Uncertainty low.

Let the GAP (between potential and cost) is defined as GAP $= k_2/k_1$ where $k_1 \leq 1 \leq k_2$ such that $\Phi(S_t) \in [k_1\text{cost}(S_t), k_2\text{cost}(S_t)], \forall S_t$ (for our other games, $k_1 = 1$). Previous work by Balcan et al. [12] established that random order updates can be helpful. Under adversarial perturbations but random order dynamics, they showed that β-nice games [3] have an expected PoU_{BR} that is at most a constant times β GAP even for constant ϵ. We establish a similar upper bound for another important class, (λ, μ)-smooth games [82] that admit a potential function Φ.

Definition 4 ((λ, μ)-smooth game [82]). *A game with a set of cost functions* cost_i *is a*

(λ, μ)-smooth game *if*

$$\textstyle\sum_i \text{cost}_i(S'_i, S_{-i}) \leq \lambda \cdot \text{cost}(S') + \mu \cdot \text{cost}(S), \forall S, S' \textit{ where } \lambda > 0 \textit{ and } \mu \in (0, 1).$$

Informally, unilateral strategy updates from state S towards another state S' cannot increase cost significantly beyond the costs of S and S'.

This class offers a unified framework for quantifying (often exactly) the price of anarchy as $\frac{\lambda}{1-\mu}$ and includes congestion games with polynomial delay functions. For example, linear congestion games are (5/3,1/3) smooth, which can be used to prove optimal bounds on the price of total anarchy [82].

Let $OPT = \min_S \text{cost}(S)$ denote the socially optimal cost. The main result of this section shows that the expected potential at any step is bounded by a state-independent factor times the initial cost. This immediately yields a bound on the expected PoU.

Theorem 11. $\mathbb{E}[\Phi(S_t)] \leq \max[5k_2 \frac{\lambda}{1-\mu} OPT, k_2 \text{cost}(S_0)] \leq k_2 \frac{5\lambda}{1-\mu} \text{cost}(S_0), \forall t$ *for* $\epsilon < \min(\frac{3-4\mu}{31}, \frac{1}{17})$ *where* $\frac{\lambda}{1-\mu} OPT$ *is an upper bound on the cost of any Nash equilibrium [82]. Hence,* $\mathbb{E}[\text{cost}(S_t)] \leq \frac{5\lambda}{1-\mu} \text{cost}(S_0) \cdot \text{GAP}$.

The rest of this section is devoted to proving this claim. We prove this by showing the potential value is either decreasing, or acceptably small at every time step. We show that if the cost at a state is low, then the potential is also low and cannot increase very much. If the cost is high, then the potential must decrease in expectation. We begin by introducing some definitions.

For a state S_t with player strategies $S_1, \ldots, S_n$, let cost^t denote the perturbed costs at this time step. Let $BR(S_{-i})$ be a player i's best-response (given true costs) to the other players' strategies S_{-i} and let $\widetilde{BR}(S_{-i})$ be i's best-response given perturbed costs.

- Let $\Delta_i(S_t)$ be player i's true cost reduction when best-responding in the current state, i.e. $\Delta_i(S_t) := \text{cost}_i(S_i, S_{-i}) - \text{cost}_i(BR(S_{-i}), S_{-i})$. Let $\Delta(S_t) = \sum_i \Delta_i(S_t)$.
- Let $\tilde{\Delta}_i(S_t) := \text{cost}^t_i(S_t) - \text{cost}^t_i(\widetilde{BR}(S_{-i}), S_{-i})$ be i's reduction in perturbed cost due to i's best-response given perturbed costs.

- Let $\hat{\Delta}_i(S_t) := \text{cost}_i(S_t) - \text{cost}_i(\widetilde{BR}(S_{-i}), S_{-i})$ be i's (real) reduction in unperturbed cost due to i's perceived best-response given perturbed costs.

Note that $\mathbb{E}_i[\tilde{\Delta}_i]$ is the expected *perceived* reduction in cost when a player best responds, and that $\mathbb{E}_i[\hat{\Delta}_i]$ is the expected reduction in cost. However, since this is an exact potential game, $-\mathbb{E}_i[\hat{\Delta}_i] = \mathbb{E}[\Phi(S_{t+1}) - \Phi(S_t)]$, precisely the expected drop in potential at time t.

Observe that

$$\begin{aligned}\hat{\Delta}_i(S_t) &= \text{cost}_i(S_t) - \text{cost}_i(\widetilde{BR}(S_{-i}), S_{-i}) \\ &\geq \text{cost}_i^t(S_t) - \epsilon\text{cost}_i(S) - \text{cost}_i^t(\widetilde{BR}(S_{-i}), S_{-i}) - \epsilon\text{cost}_i(\widetilde{BR}(S_{-i}), S_{-i}) \\ &= \tilde{\Delta}_i(S_t) - 2\epsilon\text{cost}_i(S_t) + \epsilon\hat{\Delta}_i(S_t)\end{aligned}$$

Rearranging we find that

$$\hat{\Delta}_i(S_t) \geq \frac{\tilde{\Delta}_i(S_t) - 2\epsilon\text{cost}_i(S_t)}{1-\epsilon} \tag{4}$$

In any high cost state, some players can significantly lower their cost by best-responding. Lemma 4 shows that a high cost implies that the true best response move has a large reduction in cost for some players. Lemma 5 translates that to the perceived best-responses that are actually played, to show that the potential necessarily decreases in high cost states.

Lemma 4. *At any time t, if $\text{cost}(S_t) \geq \frac{2\lambda}{1-\mu}OPT$ then $(1-\mu)\text{cost}(S_t) \leq 2\Delta(S_t)$.*

Proof. Let $S^* := \text{argmin}_S \text{cost}(S)$, be a state with cost OPT. From (λ, μ)-smoothness we get

$$\textstyle\sum_i \text{cost}_i(S_i^*, S_{-i}) \leq \lambda OPT + \mu\text{cost}(S_t)$$

A best response costs no more than the response played in OPT

$$\textstyle\sum_i \text{cost}_i(BR(S_{-i}), S_{-i}) \leq \sum_i \text{cost}_i(S_i^*, S_{-i}) \leq \lambda OPT + \mu\text{cost}(S_t),$$

We now add $\Delta(S_t)$ to both sides and multiply through by 2. Recall that $\Delta_i(S_t) + \text{cost}_i(BR(S_{-i}), S_{-i}) = \text{cost}_i^t(S_t)$. Summing over all i we see:

$$2\text{cost}(S_t) \leq 2\lambda OPT + 2\Delta(S_t) + 2\mu\text{cost}(S_t)$$

rearranging, we find

$$2(1-\mu)\mathrm{cost}(S_t) \le 2\lambda OPT + 2\Delta(S_t) \le (1-\mu)\mathrm{cost}(S_t) + 2\Delta(S_t)$$

where the last line follows from the assumption that $\mathrm{cost}(S_t) \ge \frac{2\lambda}{1-\mu}OPT$. □

Thus if the cost is high, a correct best-response decreases the cost significantly. This also implies a large drop expected in potential is large from high cost states, as our next lemma shows.

Lemma 5. *At any time t, for $\epsilon < \frac{3-4\mu}{31}$, if $\mathrm{cost}(S_t) \ge \frac{2\lambda}{1-\mu}OPT$ then $\mathbb{E}[\Phi(S_{t+1}) - \Phi(S_t)] \le -\mathrm{cost}(S_t)/(8n)$.*

Proof. We have, by Eq. (5.2) in [12], $\tilde{\Delta}_i(S_t) \ge \Delta_i(S_t) - 2\epsilon\mathrm{cost}_i(S_t)$ since the perturbations are only at most an ϵ fraction of the true cost.

Since $\mathrm{cost}(S_t)$ is large, this in turn implies that $\tilde{\Delta}_i(S_t) \ge \mathrm{cost}_i(S_t)(\frac{1-\mu}{2} - 2\epsilon)$, by applying Lemma 4.

Thus, applying Equation 4 we have $\hat{\Delta}_i(S_t) \ge (\tilde{\Delta}_i(S_t) - 2\epsilon\mathrm{cost}_i(S_t))/(1-\epsilon)$. Putting this all together we get

$$\begin{aligned}
-\mathbb{E}[\Phi(S_{t+1}) - \Phi(S_t)] &= \mathbb{E}_i[\hat{\Delta}_i(S_t)] \\
&\ge \mathbb{E}_i\left[\frac{\tilde{\Delta}_i(S_t) - 2\epsilon\mathrm{cost}_i(S_t)}{1-\epsilon}\right] \\
&\ge \mathbb{E}_i\left[\frac{\left(\frac{1-\mu}{2} - 2\epsilon\right)\mathrm{cost}_i(S_t) - 2\epsilon\mathrm{cost}_i(S_t)}{1-\epsilon}\right] \\
&= \left[\frac{\frac{1-\mu}{2} - 2\epsilon - 2\epsilon}{1-\epsilon}\right]\mathbb{E}_i[\mathrm{cost}_i(S_t)] \\
&= \left[\frac{\frac{1-\mu}{2} - 4\epsilon}{1-\epsilon}\right]\frac{\mathrm{cost}(S_t)}{n}.
\end{aligned}$$

Thus if $\epsilon < \frac{3-4\mu}{31}$, we have $\mathbb{E}[\Phi(S_{t+1}) - \Phi(S_t)] \le -\mathrm{cost}(S_t)/(8n)$, as desired □

Lemma 6. *The expected increase in potential (after an arbitrary best-response move) $\mathbb{E}[\Phi(S_{t+1}) - \Phi(S_t)]$ is at most $2\epsilon\mathrm{cost}(S_t)/(n(1-\epsilon))$.*

The proof is due to [12] and offered here for completeness.

Proof. Again, by Equation 4 for any player $\hat{\Delta}_i(S_t) \geq (\tilde{\Delta}_i(S_t) - 2\epsilon\text{cost}_i(S_t))/(1-\epsilon)$. Recall that

$$\begin{aligned}
\mathbb{E}\left[\Phi(S_{t+1}) - \Phi(S_t)\right] &= -\mathbb{E}_i\left[\hat{\Delta}\right] \\
&\leq \mathbb{E}_i\left[\frac{2\epsilon\text{cost}_i(S_t) - \tilde{\Delta}_i}{1-\epsilon}\right] \\
&\leq \mathbb{E}_i\left[\frac{2\epsilon\text{cost}_i(S_t)}{1-\epsilon}\right] \\
&= \frac{2\epsilon\text{cost}(S_t)}{(1-\epsilon)n}
\end{aligned}$$

The second step follows as $\tilde{\Delta}_i$ is non-negative. □

Finally, we prove our upper bound on the expected PoU.

Proof of Theorem 11. This proof follows the structure of a similar result on β-nice games by Balcan et al. [12]. We show that at a time step where the expected cost is low, then the expectation will remain small. On the other hand, if the expected potential is high, i.e. $\mathbb{E}[\text{cost}(S_t)] \geq 4\frac{\lambda}{1-\mu}OPT$, the potential's expectation cannot increase, i.e. $\mathbb{E}[\Phi(S_{t+1})] \leq \mathbb{E}[\Phi(S_t)]$.

First we show that if the expected cost is low on time step t, it will not increase too much in the next time step. Assume that $\mathbb{E}[\text{cost}(S_t)] \leq 4\frac{\lambda}{1-\mu}OPT$. We will show that $\mathbb{E}[\text{cost}(S_{t+1})] < 5k_2\frac{\lambda}{1-\mu}OPT$. Note that the potential is also small, $\mathbb{E}[\Phi(S_t)] \leq 4k_2\frac{\lambda}{1-\mu}OPT$. By Lemma 6 we can bound the expected increase in potential.

$$\mathbb{E}[\Phi(S_{t+1}) - \Phi(S_t)] \leq 8\epsilon\frac{\lambda}{1-\mu}\frac{OPT}{n(1-\epsilon)} \leq \frac{\lambda}{1-\mu}OPT$$

The last inequality holds provided $\epsilon < 1/9$. Thus $\mathbb{E}[\Phi(S_{t+1})] \leq (4k_2+1)\frac{\lambda}{1-\mu}OPT \leq 5k_2\frac{\lambda}{1-\mu}OPT$. We know that the potential exceeds the cost by at most a factor of k_1, and that $k_2/k_1 = \textsc{gap}$. Thus we can conclude that $\mathbb{E}[\text{cost}(S_{t+1})] \leq 5\frac{\lambda}{1-\mu}OPT \cdot \textsc{gap}$.

Now we consider the case where $\text{cost}(S_t) \geq 4\frac{\lambda}{1-\mu}OPT$. Let $p_t = \mathbb{P}\left[\text{cost}(S_t) \geq 2\frac{\lambda}{1-\mu}OPT\right]$. We have

$$\begin{aligned}\mathbb{E}[\text{cost}(S_t)] = p_t\mathbb{E}&\left[\text{cost}(S_t)\,\middle|\,\text{cost}(S_t) \geq 2\frac{\lambda}{1-\mu}OPT\right]\\ &+ (1-p_t)\mathbb{E}\left[\text{cost}(S_t)\,\middle|\,\text{cost}(S_t) \leq 2\frac{\lambda}{1-\mu}OPT\right]\\ \leq p_t\mathbb{E}&\left[\text{cost}(S_t)\,\middle|\,\text{cost}(S_t) \geq 2\frac{\lambda}{1-\mu}OPT\right] + 2\frac{\lambda}{1-\mu}OPT\end{aligned}$$

Because we are in the case where $\mathbb{E}[\text{cost}(S_t)] \geq 4\frac{\lambda}{1-\mu}OPT$, we can conclude that:

$$\mathbb{E}\left[\text{cost}(S_t)\,\middle|\,\text{cost}(S_t) \geq 2\frac{\lambda}{1-\mu}OPT\right] \geq \frac{2}{p_t}\frac{\lambda}{1-\mu}OPT.$$

We now can evaluate the change in potential

$$\begin{aligned}\mathbb{E}[\Phi(S_{t+1}) - \Phi(S_t)] \leq & p_t\mathbb{E}\left[\Phi(S_{t+1}) - \Phi(S_t)\,\middle|\,\text{cost}(S_t) \geq 2\frac{\lambda}{1-\mu}OPT\right]\\ &+ (1-p_t)\mathbb{E}\left[\Phi(S_{t+1}) - \Phi(S_t)\,\middle|\,\text{cost}(S_t) < 2\frac{\lambda}{1-\mu}OPT\right]\end{aligned}$$

We can apply Lemma 5 to the first expectation, and Lemma 6 to the second.

$$\begin{aligned}&\leq - p_t/(8n)\mathbb{E}\left[\text{cost}(S_t)\,\middle|\,\text{cost}(S_t) \geq 2\frac{\lambda}{1-\mu}OPT\right]\\ &\quad+ \frac{2\epsilon(1-p_t)}{n(1-\epsilon)}\mathbb{E}\left[\text{cost}(S_t)\,\middle|\,\text{cost}(S_t) \leq 2\frac{\lambda}{1-\mu}OPT\right]\end{aligned}$$

Recalling $\mathbb{E}\left[\text{cost}(S_t)\,\middle|\,\text{cost}(S_t) \geq 2\frac{\lambda}{1-\mu}OPT\right] \geq \frac{2}{p_t}\frac{\lambda}{1-\mu}OPT$

$$\begin{aligned}&\leq \left(\frac{-1}{8n}\right)2\left(\frac{\lambda}{1-\mu}OPT\right) + \frac{2\epsilon}{n(1-\epsilon)}\left(2\frac{\lambda}{1-\mu}OPT\right)\\ &\leq \frac{2}{n}\frac{\lambda}{1-\mu}OPT\left(\frac{-1}{8} + \frac{2\epsilon}{1-\epsilon}\right)\end{aligned}$$

which is non-positive for $\epsilon < 1/17$. □

2.7 Concluding Remarks

We quantify the long-term increase of social costs due to local uncertainty in potential games. We analyze natural dynamics, improved-response and best-response, under the weakest assumptions on the order of players updates. We use a natural model of cost perturbations, for which we obtain lower and upper bounds for important classes: consensus games (not studied before from the perspective of uncertainty), set-covering games, matroid congestion games and (λ, μ)-smooth games. We assess the effect of uncertainty via judiciously tuned parametrized constructions or in-depth analysis of costs. We provide the first tight bounds on social cost degradation during dynamics due to Byzantine players.

In conclusion, this work provides a precise picture of the long-term consequences of uncertainty or unmodeled low-order local effects on social costs in these important classes of games.

CHAPTER III

SEMISUPERVISED AND ACTIVE LEARNING OF TWO SIDED DISJUNCTIONS

We provide efficient algorithms for learning disjunctions in the semi-supervised setting under a natural regularity assumption introduced by [7]. We prove bounds on the sample complexity of our algorithms under a mild restriction on the data distribution. We also give an active learning algorithm with improved sample complexity and extend all our algorithms to the random classification noise setting.

3.1 Introduction

In many modern applications, like web-based information gathering, unlabeled data is abundant but labeling it is expensive. Consequently, there has been substantial effort in understanding and using semi-supervised learning (using large amounts of unlabeled data to augment limited labeled data) and active learning (where the algorithm itself asks for labels of carefully chosen examples with the goal of minimizing the human labeling effort) [91, 33].

Conceptually, what makes unlabeled data useful in the semi-supervised learning context [9, 91], is that for many learning problems, the natural regularities of the problem involve not only the form of the function being learned but also how this function relates to the distribution of data; for example, that it partitions data by a wide margin as in Transductive SVM [50] or that data contains redundant sufficient information as in Co-training [22]. Unlabeled data is useful in this context because it allows one to reduce the search space from the whole set of hypotheses, down to the set of hypotheses satisfying the regularity condition with respect to the underlying distribution. Such insights have been exploited for deriving a variety of sample complexity results [34, 52, 77, 9]. However, while in principle semi-supervised learning can provide benefits over fully supervised learning [9, 91], the corresponding algorithmic problems become much more challenging. As a consequence there has been a scarcity of efficient semi-supervised learning algorithms.

In this chapter we provide efficient algorithms with nearly optimal sample complexity for semi-supervised and active learning of disjunctions under a natural regularity assumption introduced in [7]. In particular we consider the so called two-sided disjunctions setting, where we assume that the target function is a monotone disjunction satisfying a margin like regularity assumption[1]. In the simplest case resolved in [7], the notion of "margin" is as follows: every variable is either a positive indicator for the target function (i.e., the true label of any example containing that variable is positive) or a negative indicator (i.e., the true label of any example containing that variable is negative), and no example contains both positive and negative indicators. In this work, we consider the much more challenging setting left open in [20] where *non-indicators* or *irrelevant variables*, i.e., variables that appear in both positive and negative examples, are also present.

In the semi-supervised learning setting, we present an algorithm that finds a consistent hypothesis that furthermore is compatible (in the sense that it satisfies our regularity assumption). This algorithm is proper (it outputs a disjunction), has near-optimal labeled data sample complexity provided that each irrelevant variable appears with non-negligible probability, and it is efficient when the number of irrelevant variables is $O(\log n)$. We next present a non-proper algorithm that PAC learns two-sided disjunctions with nearly the same sample complexity and whose running time is polynomial for any k.

In the active learning setting, we present an efficient active learning algorithm for two-sided disjunctions. This algorithm outputs a consistent, compatible hypothesis, with sample complexity linear in the number of irrelevant variables and independent of the probability of irrelevant variables appearing, a quantity that appears in the bounds in the semi-supervised setting. Combined with the **NP**-hardness result we show for two-sided disjunctions (see Section 3.8), the algorithm also shows that the active query ability can help to overcome computational difficulty.

We also discuss how our algorithms can be adapted to deal with random classification noise.

Discussion: To see why the presence of irrelevant variables significantly complicates the

[1]See Section 3.10 for further discussion

algorithmic problem, note that in the absence of non-indicators (the case studied in [7]), we could construct an approximation of the so called commonality graph, defined on n vertices (one per variable), by putting an edge between two vertices i and j if there is any example x in our unlabeled sample with x_i, x_j set to 1. If the target function indeed satisfies the regularity assumption, then no component will get multiple labels, so all we need to learn is a labeled example in each component. Furthermore, if the number of components in the underlying graph is small, then both in the semi-supervised and active learning setting we can learn with many fewer labeled examples then in the supervised learning setting.

Introducing non-indicators into the target concept complicates matters, because components can now have multiple labels. We could think of the non-indicators as forming a vertex cut in the commonality graph separating variables corresponding to positive indicators from those corresponding to negative ones. To learn well, one could try to find such a cut with the necessary properties to ensure compatibility with the unlabeled data (i.e. no examples are composed only of non-indicators). Unfortunately, this is a difficult combinatorial problem in general. Interestingly, we will be able to find such cut for $k = O(\log n)$ and for general k we will be still be able to learn with nearly optimal rates, if each non-indicator appears with non-negligible probability; we do this by identifying a superset of non-indicators and carefully making inferences using it. Furthermore, since classification mistakes reveal vertices in both sides of the cut, our adaptive query ability in the active learning model will allow us to actively search for vertices in the cut.

Related work: While several semi-supervised learning methods have been introduced [25, 90, 51], much of the theoretical work has focused either on sample complexity (e.g., [34, 52, 77]) or on providing polynomial time algorithms with error bounds for surrogate losses only (e.g., [78]). The few existing results with guarantees on the classification error loss hold under very stringent conditions about the underlying data distribution (e.g., independence given the label [22]). By contrast, we provide (PAC-style) polynomial time algorithms for learning disjunctions with general guarantees on the classification error loss.

We note that while a lot of the research on active learning [32, 6, 45, 46, 35, 19, 58] has *not* made an explicit regularity assumption as in semi-supervised learning, this is an interesting direction to study. As our results reveal, active learning could help overcome computational hardness limitations over (semi-supervised) passive learning in these settings.

3.2 Preliminaries and Notation

Let $X = \{0,1\}^n$ be the instance space, $Y = \{-1,1\}$ be the label set, and D denote any fixed probability distribution over X. Following [7], a two-sided disjunction h is defined as a pair of monotone disjunctions[2] (h_+, h_-) such that $h_+(x) = -h_-(x)$ for all $x \sim D$, and h_+ is used for classification. Let the concept class C be the set of all pairs[3] of monotone disjunctions and for any hypothesis $h = (h_+, h_-) \in C$, define $h(x) = h_+(x)$.

For a two-sided disjunction (h_+, h_-), variables included in h_+ are the *positive indicators*, and variables in h_- are *negative indicators*. Variables appearing neither in h_+ nor in h_- are called *non-indicators*, as the value of any such variable has no effect on the label of any example. To simplify the discussion, we will often identify binary strings in $X = \{0,1\}^n$ with subsets of the variables $V = \{x_1, \ldots, x_n\}$. In other words, we say an example x contains x_i if the i-th coordinate of x is set to 1. This allows us to speak of variables "appearing in" or "being present in" examples rather than variables being set to 1. We will use similar language when referring to hypotheses, so that a two-sided disjunction $h = (h_+, h_-)$ consists of a set h_+ of positive indicators and a set h_- of negative indicators (which completely determine a third set of non-indicators).

In the semi-supervised learning setting, we will assume that both labeled examples L and unlabeled examples U are drawn i.i.d. from D and that examples in L are labeled by the target concept h^*, where h^* is a two-sided disjunction with at most k non-indicators. We will let $|L| = m_l$ and $|U| = m_u$; both m_l and m_u will be polynomial throughout this chapter. In the active setting, the algorithm first receives a polynomially sized unlabeled sample U and it can adaptively ask for the label $\ell(x) = h^*(x)$ of any example $x \in U$. In the random classification noise model (studied in Section 3.9) we assume that the label of each labeled example is flipped with probability α.

[2]Recall that a monotone disjunction is an OR function of positive literals only, e.g. $h(x) = x_1 \vee x_3 \vee x_4$.
[3]Although we are actually interested in learning a single monotone disjunction, we need to associate each disjunction with a second disjunction in order to test compatibility.

The generalization error of a hypothesis h is given by $\mathsf{err}(h) = \Pr_{x\sim D}[h(x) \neq h^*(x)]$, the probability of h misclassifying a random example drawn from D. For a set L of labeled examples, the empirical error is given by $\mathsf{err}_L(h) = |L|^{-1} \sum_{x\in L} I[h(x) \neq h^*(x)]$. If $\mathsf{err}_L(h) = 0$ for some h we say that h is *consistent* with the data.

To formally encapsulate the regularity or compatibility assumption for two-sided disjunctions described in the introduction, we consider the regularity or *compatibility function* χ: $\chi(h, x) = I[h_+(x) = -h_-(x)]$ for any hypothesis h and example $x \in X$. In addition, we define (overloading notation) the compatibility between h and the distribution D as $\chi(h, D) = \mathrm{E}_{x\sim D}[\chi(h, x)] = \Pr_{x\sim D}[h_+(x) = -h_-(x)]$. For a set U of unlabeled examples, define the empirical compatibility between h and U as $\chi(h, U) = |U|^{-1} \sum_{x\in U} I[h_+(x) = -h_-(x)]$. If $\chi(h, U) = 1$ we say that h is *compatible* with the data. Thus a hypothesis is consistent and compatible with a set of examples if every example contains exactly one type of indicator and every labeled example contains an indicator of the same type as its label. We will assume throughout this chapter that the target function is compatible.

We define, for any $\epsilon > 0$, the reduced hypothesis class $C_{D,\chi}(\epsilon) = \{h \in C : 1-\chi(h, D) \leq \epsilon\}$, the set of hypotheses with "unlabeled error" at most ϵ. Similarly, for an unlabeled sample U, we define $C_{U,\chi}(\epsilon) = \{h \in C : 1 - \chi(h, U) \leq \epsilon\}$. The key benefit of using unlabeled data and our regularity assumption is that the number of labeled examples will only depend on $\log(C_{D,\chi}(\epsilon))$ which for helpful distributions will be much smaller than $\log(C)$.

3.2.1 The Commonality Graph

The basic structure used by all of our algorithms is a construct we call the *commonality graph*. As mentioned in the introduction, the commonality graph is the graph on variables that contains an edge between any two vertices if the corresponding variables appear together in a common example. That is, given the set U of unlabeled examples, define the commonality graph $G_{\mathrm{com}}(U) = (V, E)$ where $V = \{x_1, \ldots, x_n\}$ and E contains an edge (x_i, x_j) if and only if there is some $x \in U$ such that x_i and x_j are both set to 1 in x. Furthermore, given the set L of labeled examples, let V_L^+ be the set of variables appearing in positive examples and V_L^- be the set of variables appearing in negative examples.

The edge structure of the commonality graph and the labeled examples will allow us to draw inferences about which vertices in the graph correspond to positive indicators, negative indicators, and non-indicators in the target concept. Any variable that appears in a labeled example cannot be an indicator of the type opposite of the label. In addition, an edge between two variables implies they cannot be indicators of different types. This means that any path in the commonality graph between positive and negative indicators must contain a non-indicator. Similarly, paths that pass only through indicator variables can be used to propagate labels to the unlabeled examples.

3.3 Semi-supervised Learning

Our general strategy is to identify non-indicators and remove them from the commonality graph, reducing this problem to the simpler case. Notice that each non-indicator that appears in the unlabeled data is significant; failing to identify it can lead to incorrect inferences about a large probability mass of examples. A variable is obviously a non-indicator if it appears in both positive and negative examples. A naïve approach would be to draw enough labeled examples so that every significant non-indicator appears in examples with both labels. The problem with this approach is that some non-indicator can appear much more frequently in positive examples than in negative examples. In this case the number of examples needed by the naïve approach is inversely proportional to the probability of that non-indicator appearing in negative examples. This sample complexity can be worse than in the fully supervised case.

In our approach, it is enough to ensure each non-indicator appears in a labeled example, but not necessarily in both positive and negative examples. The number of examples needed in this case will now depend on the minimum probability of a non-indicator appearing. This allows the sample complexity to be significantly smaller than that of the naïve approach; for example, when a non-indicator appears in positive examples with constant probability while in negative examples with probability ϵ/n.

Our approach can still identify non-indicators, now by examining paths in the commonality graph. In paths whose interior vertices appear only in unlabeled examples (i.e.

are indicators) and whose endpoints appear in oppositely labeled examples, one of the endpoints must be a non-indicator. When $k = O(\log n)$ we can enumerate over all consistent compatible hypotheses efficiently by restricting our attention to a small set of paths.

If the number of non-indicators is larger, we can still find a good hypothesis efficiently by finding the non-indicators one at a time. Each time our working hypothesis makes a mistake this reveals a path whose endpoint is a non-indicator.

The number of labeled examples we require will depend on the *minimum non-indicator probability* defined by

$$\epsilon_0(D, h^*) = \min_{x_i \notin h^*_+ \cup h^*_-} \Pr_{x \sim D}[x_i = 1].$$

For notational convenience denote it simply by ϵ_0 without ambiguity. To guarantee with high probability that each non-indicator appears in some labeled example, it suffices to use $\tilde{O}(\frac{1}{\epsilon_0} \log k)$ labeled examples.

3.3.1 Finding a Consistent, Compatible Hypothesis Efficiently when $k = O(\log n)$

We now give an algorithm, along with some intuition, for finding a two-sided disjunction that is consistent and compatible with a given training set. We note that this problem is **NP**-hard in general (see Section 3.8). Given example sets L and U, the algorithm begins by constructing the commonality graph $G = G_{\text{com}}(U)$ and setting G to $G \setminus (V_+ \cap V_-)$. This removes any variables that appear in both positive and negative examples as these are guaranteed to be non-indicators.

To identify the rest of the non-indicators, we consider a new graph. Using $u \leftrightarrow_G v$ to denote the existence of a path in the graph G between vertices u and v, we define the *indicator graph* $G_{\text{ind}}(G, V_+, V_-)$ to be the bipartite graph with vertex set $V_+ \cup V_-$ and edge set $\{(u, v) \in V_+ \times V_- : u \leftrightarrow_{G \setminus (V_+ \cup V_-)} v\}$. The key idea is that an edge in this graph implies that at least one of its endpoints is a non-indicator, since the two variables appear in oppositely labeled examples but are connected by a path of indicators.

Note that the target set of non-indicators must form a vertex cover in the indicator graph. By iterating over all minimal vertex covers, we must find a subset of the target non-indicators whose removal disconnects positive examples from negative examples, and

Algorithm 2 Finding a consistent compatible two-sided disjunction

```
**Input:** unlabeled set $U$, labeled set $L$
Set $G = G_{\text{com}}(U)$, $V_+ = V_L^+$, $V_- = V_L^-$
Set $G = G \setminus (V_+ \cap V_-)$
Set $V_+ = V_+ \cap G$, $V_- = V_- \cap G$
Set $G_I = G_{\text{ind}}(G, V_+, V_-)$
**for** each minimal vertex cover $S$ of $G_I$ **do**
  Set $G' = G \setminus S$, $V'_+ = V_+ \setminus S$, $V'_- = V_- \setminus S$
  Set $h_+ = \{v \in G' : \exists u \in V'_+,\ u \leftrightarrow_{G'} v\}$
  **if** $(h_+, G' \setminus h_+)$ is consistent and compatible **then**
    **break**
  **end if**
**end for**
**Output:** hypothesis $h = (h_+, G' \setminus h_+)$
```

this corresponds to a consistent compatible hypothesis. The algorithm is summarized in Algorithm 2.

The key step in Algorithm 2 is enumerating the minimal vertex covers of the indicator graph G_I. One way to do this is as follows. First find a maximum matching M in G_I, and let m be the number of disjoint edges in M. Enumerate all 3^m subsets of vertices that cover M (for every edge in M, one or both of the endpoints can be included in the cover). For each such cover S, extend S to a minimal vertex cover of G_I by adding to S every variable not covered by M that has no neighbors already in S. This extension can always be done uniquely, so there is a one-to-one correspondence between covers of M and minimal vertex covers of G_I.

This enumeration method gives us both a concrete way to implement Algorithm 2 and a way to bound its running time. We prove in Theorem 12 that given enough data, Algorithm 2 correctly outputs a consistent compatible hypothesis with high probability. We then bound its time and sample complexity.

Theorem 12. *For any distribution D over $\{0,1\}^n$ and target concept $h^* \in C$ such that $\chi(h^*, D) = 1$, h^* has at most k non-indicators, and the minimum non-indicator probability is ϵ_0, if $m_u \geq \frac{1}{\epsilon}\left[\log \frac{2|C|}{\delta}\right]$ and*

$$m_l \geq \max\left\{\frac{1}{\epsilon_0}\log\frac{k}{\delta}, \frac{1}{\epsilon}\left[\log\frac{2|C_{D,\chi}(\epsilon)|}{\delta}\right]\right\}$$

then with probability at least $1 - 2\delta$, Algorithm 2 outputs a hypothesis $h \in C$ such that $\mathsf{err}_L(h) = 0$, $\chi(h, U) = 1$, and $\mathsf{err}(h) \leq \epsilon$. Furthermore, when $k = O(\log n)$ the algorithm runs in time at most poly(n).

Proof. We separate the proof into three sections, first proving consistency and compatibility of the output hypothesis, then giving the sample sizes required to guarantee good generalization, and finally showing the overall running time of the algorithm.

Consistency and Compatibility: The exit condition for the loop in Algorithm 2 guarantees that the algorithm will output a consistent compatible hypothesis, so long as a suitable minimal vertex cover of G_I is found. Thus, it suffices to show that such a vertex cover exists with high probability when L is large enough.

By the definition of ϵ_0 along with the independence of the samples and a union bound, if $m_l > \frac{1}{\epsilon_0} \log \frac{k}{\delta}$, then with probability at least $1 - \delta$, all non-indicator variables appear in some labeled example. We will assume in the remainder of the proof that all variables not in $V_L^+ \cup V_L^-$ are indicators.

Since an edge in G between indicators forces both endpoints to be of the same type, a path through indicators does the same. Edges in G_I correspond to such paths, but the endpoints of such an edge cannot be indicators of the same type because they appear in differently labeled examples. It follows that at least one endpoint of every edge in G_I must be a non-indicator.

Now let V_0 be the set of non-indicators in the target hypothesis. The above observations imply that V_0 contains a vertex cover of G_I. It follows that there must exist a subset $\tilde{S} \subseteq V_0$ that is a minimal vertex cover of G_I. Let $\tilde{h} = (\tilde{h}_+, \tilde{h}_-)$ be the hypothesis h formed from the minimal vertex cover $S = \tilde{S}$. We only need to show that $\tilde{h}$ is both consistent and fully compatible.

Every indicator of h^* is also an indicator of $\tilde{h}$ since only true non-indicators were removed from G and all remaining variables became indicators in $\tilde{h}$. Since every example contains an indicator of h^*, every example must contain an indicator of $\tilde{h}$ of the correct type. Furthermore, if an example contained both positive and negative indicators, this would imply an edge still present in G_I. But removing a vertex cover removes all edges, so this is impossible. Hence $\tilde{h}$ is a consistent, fully compatible hypothesis.

Generalization Error: If $m_u \geq \frac{1}{\epsilon}[\log \frac{2|C|}{\delta}]$ and $m_l \geq \max\{\frac{1}{\epsilon_0} \log \frac{k}{\delta}, \frac{1}{\epsilon}[\log \frac{2|C_{D,\mathcal{X}}(\epsilon)|}{\delta}]\}$, the above analysis states that Algorithm 2 will fail to produce a consistent compatible hypothesis with probability at most δ. Furthermore, an algorithm with true error rate greater than ϵ will be fully consistent with a labeled set of size m_l with probability at most $\delta / C_{D,\mathcal{X}}(\epsilon)$. Union bounding over all compatible hypotheses we see that a consistent compatible hypothesis will fail to have an error rate less than ϵ with probability at most δ. By a union bound over the two failure events, the overall probability of failure is $\leq 2\delta$.

Running Time: Since checking consistency and compatibility can be done in time polynomial in the number of examples, the limiting factor in the running time is the search over minimal vertex covers of G_I. In a bipartite graph, the size of the minimum vertex cover is equal to the size of the maximum matching. The set of k non-indicators in the target hypothesis includes a vertex cover of G_I, so the size m of the maximum matching is at most k. There is one minimal vertex cover for each of the 3^m covers of the maximum matching, so the number of minimal vertex covers to search is at most 3^k. □

3.3.2 A General Semi-supervised Algorithm

Algorithm 2 is guaranteed (provided the labeled set is large enough) to find a hypothesis that is both consistent and compatible with the given data but is efficient only when k is logarithmic in n. When k is instead polylogarithmic in n, our algorithm is no longer efficient but still achieves a large improvement in sample complexity over supervised learning. We now present an efficient algorithm for finding a low-error (but not necessarily consistent and compatible) hypothesis which matches the sample complexity of Algorithm 2.

The algorithm, summarized in Algorithm 3, begins by constructing the commonality graph from the unlabeled examples and identifying potential indicators from a subset of the labeled examples. We use sample(m, S) to denote a random sample of m elements from set S. An initial hypothesis is built and tested on the sequence of remaining labeled examples. If the hypothesis makes a mistake, it is updated and testing continues. Each update corresponds to either identifying a non-indicator or labeling all indicators in some

Algorithm 3 Learning a Low-error Hypothesis for Two-Sided Disjunctions

```
Input: data sets $U$ and $L$, parameters $\epsilon$, $\delta$, $k$
Set $L' = \text{sample}(\frac{1}{\epsilon_0} \log \frac{k}{\delta}, L)$ and $L = L \setminus L'$
Set $G = G_{\text{com}}(U) \setminus (V_{L'}^{+} \cap V_{L'}^{-})$
Set $P = G \cap (V_{L'}^{+} \cup V_{L'}^{-})$
Set $h = h_{G,L'}$ and $c = 0$
while $L \neq \emptyset$ and $c \leq \frac{1}{\epsilon} \log \frac{k+T}{\delta}$ do
  Set $x = \text{sample}(1, L)$
  Set $L = L \setminus \{x\}$, and $L' = L' \cup \{x\}$
  if $h(x) \neq \ell(x)$ then
    Set $G = G \setminus \text{nn}_{G,P}(x)$
    Set $h = h_{G,L'}$ and $c = 0$
  else
    Set $c = c + 1$
  end if
end while
Output: the hypothesis $h$
```

connected component in the commonality graph, so the number of updates is bounded. Furthermore, if the hypothesis makes no mistakes on a large enough sequence of consecutive examples, then with high probability it has a small error rate overall. This gives us a stopping condition and allows us to bound the number of examples seen between updates.

The hypotheses in Algorithm 3 use a variation on nearest neighbor rules for classification. Given a commonality graph G and a set L of labeled examples, the associated nearest neighbor hypothesis $h_{G,L}$ classifies an example x the same as the nearest labeled example in L. The distance between two examples x and x' is the measured by the minimum path distance in G between the variables in x and the variables in x'. If no examples in L are connected to x, then $h_{G,L}$ classifies x negative by default. For convenience, we use $\text{nn}_{G,S}(x)$ to denote the vertex in the set S nearest to a variable in the example x via a path in G. If no such vertex exists, $\text{nn}_{G,S}(x)$ returns the empty set. Using hypotheses of this form ensures that the neighbor variable used to classify an example x is connected to x by a path through indicators, which allows us to propagate its label to the new example. If the example is misclassified, we must have been fooled by a non-indicator.

The number of examples used by Algorithm 3 depends on T, the number of connected components in the commonality graph after removing all non-indicators. Lemma 8 bounds this quantity by the number of compatible hypotheses. The proof is left to Section 3.9.

Lemma 7. *Let G be the graph that results from removing all non-indicators from $G_{\text{com}}(U)$, and suppose G is divided into T connected components. If $m_u \geq \frac{2n^2}{\epsilon} \log \frac{n}{\delta}$, then $T \leq \log_2 |C_{D,\chi}(\epsilon)|$ with probability at least $1 - \delta$.*

The following theorem bounds the number of examples sufficient for Algorithm 3 to output a low-error hypothesis.

Theorem 13. *For any distribution D over $\{0,1\}^n$ and target concept $h^* \in C$ such that $\chi(h^*, D) = 1$, h^* has at most k non-indicators, and the minimum non-indicator probability is ϵ_0, if $m_u \geq \frac{2n^2}{\epsilon} \log \frac{n}{\delta}$ and*

$$m_l \geq \frac{1}{\epsilon_0} \log \frac{k}{\delta} + \frac{k + \log |C_{D,\chi}(\epsilon)|}{\epsilon} \left[\log \frac{k + \log |C_{D,\chi}(\epsilon)|}{\delta} \right]$$

then with probability at least $1 - 3\delta$, Algorithm 3 outputs a hypothesis h in polynomial time such that $\mathsf{err}(h) \leq \epsilon$.

Proof. **Generalization Error:** First note that according to the loop exit condition, Algorithm 3 outputs the first hypothesis it encounters that correctly classifies a sequence of at least $\frac{1}{\epsilon} \log \frac{k+T}{\delta}$ i.i.d. examples from D. If $\mathsf{err}(h) > \epsilon$ for some hypothesis h, then the probability that h correctly classifies such a sequence of examples is at most $(1-\epsilon)^{\frac{1}{\epsilon} \log \frac{k+T}{\delta}} \leq \frac{\delta}{k+T}$. Assuming Algorithm 3 updates its hypothesis at most $k + T$ times, a union bound over the $k + T$ hypotheses considered guarantees that with probability at least $1 - \delta$, the hypothesis output by Algorithm 3 has error rate at most ϵ. In the remainder of the proof, we will bound the total number of samples required by Algorithm 3 and show that it makes at most $k + T$ updates to its hypothesis.

Mistake Bound: By the definition of ϵ_0, the initial set of m_l labeled examples ensures that with probability at least $1 - \delta$ all non-indicators are included in the potential indicator set P, so all variables outside P (call this set Q) are indicators. We will assume such an event holds throughout the remainder of the proof. In particular, this means that any paths through Q must consist entirely of indicators of the same type.

Suppose at some point during the execution of Algorithm 3, the intermediate hypothesis h misclassifies an example x. There are two types of such mistakes to consider. If the variables in x are not connected to any variables in P, then by the above observation, all variables connected to x are indicators of the same type, and in particular, they are indicators of the type corresponding to the label of x. This means that this type of mistake can occur only when h knows of no labeled examples connected to x. Once h is updated to be $h_{G,L'}$ where $x \in L'$, h can make no further mistakes of this type on any examples connected to x. Thus, Algorithm 3 can make at most T mistakes of this type before all connected components have labeled examples.

The hypothesis $h_{G,L'}$ labels x with the label of the example of L' containing $\mathrm{nn}_{G,P}(x)$. If x is labeled incorrectly, then this must be an example with label opposite that of x. But since the path between $\mathrm{nn}_{G,P}(x)$ and x consists only of vertices not in P, i.e. indicators, we conclude that $\mathrm{nn}_{G,P}(x)$ must be a non-indicator. Algorithm 3 can make at most k mistakes of this type before all non-indicators are removed from G.

Sample Complexity and Running Time: We have shown that after Algorithm 3 makes $k + T$ updates, all non-indicators have been removed from G and all connected components in G contain a variable that has appeared in a labeled example. Since at most $\frac{1}{\epsilon} \log \frac{k+T}{\delta}$ examples can be seen between updates, the total number of labeled examples needed by Algorithm 3 is at most

$$\frac{1}{\epsilon_0} \log \frac{k}{\delta} + \frac{k+T}{\epsilon} \log \frac{k+T}{\delta}.$$

Straightforward algebra and an application of Lemma 8 yields the bound given in the theorem statement, and a union bound over the three failure events of probability δ yields the stated probability of success. The time complexity is clearly polynomial in n per example and therefore polynomial overall. $\square$

3.4 Active Learning

We now consider the problem of learning two-sided disjunctions in the active learning model, where the learner has access to a set U of unlabeled examples and an oracle that returns the label of any example in U we submit. The additional power provided by this model allows us to use the same strategy as in the semi-supervised algorithm in Section 3.3.2 while achieving sample complexity bounds independent of ϵ_0.

As in Section 3.3.2, the goal will be to identify and remove non-indicators from the commonality graph and obtain labeled examples for each of the connected components in the resulting graph. In the semi-supervised model we could identify a mistake when there was a path connecting a positive labeled example and a negative labeled example. To identify non-indicators we guaranteed that they would lie on the endpoints of these labeled paths. In the active learning setting, we are able to check the labels of examples along this path, and thus are able to remove our dependence on minimum non-indicator probability parameter.

The algorithm we propose can be seen as a slight modification of Algorithm 3. The idea is to maintain a set of at least one labeled example per connected component and to test the corresponding nearest neighbor hypothesis on randomly chosen examples. If the hypothesis misclassifies some example, it identifies a path from the example to its nearest neighbor. Since these examples have opposite labels, a non-indicator must be present at a point on the path where positive indicators switch to negative indicators, and such a non-indicator can found in logarithmically many queries by actively choosing examples to query along this path in a binary search pattern. The search begins by querying the label of an example containing the variable at the midpoint of the path. Depending on the queried label, one of the endpoints of the path is updated to the midpoint, and the search continues recursively on the smaller path whose endpoints still have opposite labels. Let $\text{BinarySearch}_{G,L}(x)$ return the non-indicator along the path in G from a variable in x to $\text{nn}_{G,L}(x)$. As with Algorithm 3, the algorithm halts after removing all k non-indicators or after correctly labeling a long enough sequence of random examples.

The details are described in Algorithm 4, and the analysis is presented in Theorem 14.

Theorem 14. *For any distribution D over $\{0,1\}^n$ and target concept $h^* \in C$ such that $\chi(h^*, D) = 1$ and h^* has at most k non-indicators, let T be the number of connected components in the graph G that results from removing all non-indicators from $G_{\text{com}}(U)$. If*

Algorithm 4 Actively Learning Two-Sided Disjunctions

```
**Input:** unlabeled data $U$, parameters $\epsilon$, $\delta$, $k$
Set $G = G_{\text{com}}(U)$ and $L = \emptyset$
**for** each connected component $R$ of $G$ **do**
  Choose $x \in U$ such that $x \subseteq R$
  Set $L = L \cup \{(x, \ell(x))\}$
**end for**
Set $h = h_{G,L}$ and $c = 0$
**while** $c \leq \frac{1}{\epsilon} \log \frac{k}{\delta}$ **do**
  Set $x = \text{sample}(1, U)$ and $L = L \cup \{(x, \ell(x))\}$
  **if** $h(x) \neq \ell(x)$ **then**
    Set $v = \text{BinarySearch}_{G,L}(x)$
    Set $G = G \setminus \{v\}$
    **for** each new connected component $R$ of $G$ **do**
      Choose $x \in U$ such that $x \subseteq R$
      Set $L = L \cup \{(x, \ell(x))\}$
    **end for**
    Set $h = h_{G,L}$ and $c = 0$
  **else**
    Set $c = c + 1$
  **end if**
**end while**
**Output:** the hypothesis $h$
```

$m_u \geq \frac{2n^2}{\epsilon} \log \frac{n}{\delta}$ *then after at most*

$$m_q = O\left(\log |C_{D,\mathcal{X}}(\epsilon)| + k\left[\log n + \frac{1}{\epsilon} \log \frac{k}{\delta}\right]\right)$$

label queries, with probability $\geq 1 - 2\delta$*, Algorithm 4 outputs a hypothesis* h *in polynomial time such that* $\mathsf{err}(h) \leq \epsilon$.

Proof. **Generalization Error:** According to the exit condition of the loop in Step 3, Algorithm 4 outputs the first hypothesis it encounters that correctly classifies a sequence of at least $\frac{1}{\epsilon} \log \frac{k}{\delta}$ i.i.d. examples from D. If $\mathsf{err}(h) > \epsilon$ for some hypothesis h, then the probability that h correctly classifies such a sequence of examples is at most $(1-\epsilon)^{\frac{1}{\epsilon} \log \frac{k}{\delta}} \leq \frac{\delta}{k}$. Assuming Algorithm 4 updates its hypothesis at most k times, a union bound over the k hypotheses considered guarantees that with probability at least $1 - \delta$, the hypothesis output by Algorithm 4 has error rate at most ϵ. In the remainder of the proof, we will bound the total number of samples required by Algorithm 4 and show that it makes at most k updates to its hypothesis.

Queries per Stage: In the loops over connected components of G, one label is queried for each component. The components are those formed by removing from G a subset of the non-indicators, so the total number of queries made in these loops is at most T, the number of components after removing all non-indicators.

Now suppose the hypothesis h misclassifies an example x. Let x' be the nearest labeled example to x, and let x_i and x_j be the endpoints of the shortest path from x to x' in G. If each variable along the path appears in examples of only one label, then there could be no path between x_i and x_j, which appear in examples with different labels. Thus, there must exist a variable along the path from x_i to x_j that appears in both positive and negative examples, i.e. a non-indicator. Since the commonality graph was constructed using the examples in U, we can query the labels of examples that contain variables between x_i and x_j in order to find the non-indicator. Using binary search, the number of queries is logarithmic in the path length, which is at most n.

Query Complexity and Running Time: Each mistake results in removing a non-indicator from the G, so at most k mistakes can be made. For each mistake, $O(\log n)$ queries are needed to find a non-indicator to remove and at most $\frac{1}{\epsilon}\log\frac{k}{\delta}$ more queries are used before another mistake is made. Combined with the queries for the connected components, we can bound the total number of queries by $O\left(T + k\left[\log n + \frac{1}{\epsilon}\log\frac{k}{\delta}\right]\right)$. We can further bound T by $\log|C_{D,\mathcal{X}}(\epsilon)|$ via Lemma 8, and pay the price of an additional δ probability of failure. The running time for this algorithm is clearly polynomial. □

3.5 Random Classification Noise

We now consider the more challenging problem of learning two-sided disjunctions under random classification noise with noise rate $\alpha \in [0, 1/2\}$. As before, the underlying target concept can be represented by a two-sided disjunction, but unlike the rest of this work, the observed labels are noisy. Specifically, the observed label of each example is reversed (from that given by the target concept) independently with probability α. Note that compatibility relies only on unlabeled data, which does not change in the face of noise.

All our previous algorithms can be extended to this noisy setting. Since we can still build an accurate commonality graph, we only need to ensure that we can still determine the indicator type of each variable, which can be done by taking a majority vote of the labels of those examples containing the variable. To guarantee with high probability that we will not be fooled by noise, we can use this majority voting scheme only for variables that appear in at least $\tilde{O}(\frac{1}{(1-2\alpha)^2})$ labeled examples, a property we call *significance*. For semi-supervised learning, we need more labeled data up front to ensure that we have enough significant variables. In the active setting, we can selectively make variables significant by querying the labels of enough relevant unlabeled examples.

Some additional technical modifications are required for the same guarantees to hold. We discuss some of these details below for the extensions to Algorithms 3 and 4 and leave the theorems and proofs to Section **??**. While Algorithm 2 also extends to this setting, its main advantage was obtaining consistent and compatible hypotheses, which is no longer achievable in the noisy setting.

Semi-supervised Learning: We first need enough examples to ensure that all non-indicators are significant. We then build a hypothesis in a similar manner to the noise-free setting by deciding indicator types via majority voting. To verify the hypothesis, we test it on an entire set of labeled examples instead of stopping to update at the first mistake. If the empirical error is small, we are guaranteed to have a hypothesis with small error. Otherwise, the high error may be caused either by a component without any labeled variables (which can be corrected by a majority vote of the examples in that component) or a non-indicator. In the latter case, we need to distinguish between the non-indicator causing an error rate of $\alpha + O(\epsilon(1-2\alpha)/k)$ and other significant indicators causing an error rate of α, and this requires $\tilde{O}\left(\frac{k^2}{(1-2\alpha)^2\epsilon^2}\right)$ labeled examples. As in the noise free case, the hypothesis is updated at most $k+T$ times, which leads to a label complexity of $\tilde{O}\left(\frac{k+T}{(1-2\alpha)^2}\left[\frac{1}{\epsilon_0} + \frac{k^3}{\epsilon^3} + \frac{T}{\epsilon}\right]\right)$.

Active Learning: There are two main alterations to make to Algorithm 4 in order to extend it to this setting. First, we use majority voting to determine indicator types instead of using single examples. Second, instead of performing binary search over vertices in the path that connects positive and negative examples, we need to search over edges in order to distinguish disagreement caused by non-indicators from that caused by noise. If an edge contains an indicator, a majority vote of examples containing the two variables in the edge will agree with the type of the indicator, even if the edge contains a non-indicator. Any variable contained in edges of different labels must be a non-indicator. This leads to a query complexity of $\tilde{O}\left(\frac{1}{(1-2\alpha)^2}\left[T + \frac{k^2}{\epsilon} + \frac{k}{\epsilon^2}\right]\right)$.

3.6 Distributed Learning

The above algorithms can be adapted to the distributed learning model of Balcan et al. [10] in which the data is assumed to be distributed among K entities. In particular, for $1 \leq i \leq K$, entity i has access to examples drawn i.i.d. from distribution D_i, and the goal is to find a hypothesis that has a low error rate on the joint mixture of $D_1, \ldots, D_K$ with as little inter-entity communication as possible. Balcan et al. study this model in the fully supervised setting and are concerned with finding protocols for various settings with optimal communication complexity while only requiring that each entity uses a polynomial number of samples. A modification of this model that applies to the semi-supervised setting is to attempt to simultaneously optimize communication complexity and label complexity while ensuring that each entity uses a polynomial amount of unlabeled data.

Our goal is to analyze the communication and sample complexity of learning two-sided disjunctions in this semi-supervised distributed learning model. Table **??** summarizes our findings. For comparison, we first analyze the sample complexity of two baseline algorithms given in [10] that apply to learning disjunctions in general with no unlabeled data. We then adapt the SSL algorithms for the case of two-sided disjunctions with no non-indicators in two ways: one that achieves the same overall label complexity as in the non-distributed setting and one that achieves a better communication complexity while sacrificing label complexity in some cases. The main result of this section is an adaptation of Algorithm 3 that achieves the same overall label complexity in the distributed setting and incurs a communication cost of only $O(Kn \log n)$ bits. It is interesting to note that this algorithm allows the labeled examples to be evenly distributed among all K entities, so each entity needs a multiplicative factor of K fewer labels than an entity running Algorithm 3 on its own.

Setting	Communication complexity	Label complexity (per entity)
Baseline 1	$O(Kn)$ bits	$O\left(\frac{n}{\epsilon}\log\frac{K}{\delta}\right)$
Baseline 2	$O(n\log n)$ bits	$O\left(\frac{n}{K\epsilon}\log\frac{n}{\delta}\right)$
$k=0$ Alg. 1	$O(Kn)$ bits	$O\left(\frac{T_i}{\epsilon}\log\frac{KT_i}{\delta}\right)$
$k=0$ Alg. 2	$O(Kn\log n)$ bits	$O\left(\frac{T}{K\epsilon}\log\frac{T}{\delta}\right)$
$k\geq 0$ Alg. 1	$O(Kn\log n)$ bits	$O\left(\frac{k+T}{K\epsilon}\log\frac{k+T}{\delta}\right)$

Table 1: Summary of results for learning two-sided disjunctions in the semi-supervised distributed learning model.

3.6.1 Baseline Algorithms

One of the most basic algorithms for learning disjunctions in the fully supervised distributed model is a special case of the more general algorithm for intersection- or union-closed classes given in [10]. In particular, the class of disjunctions can be learned by inverting each example and its label and running the algorithm for learning conjunctions. Using this approach, each entity i needs only $(n/\epsilon)\log(K/\delta)$ examples to learn a hypothesis h_i with at most ϵ error on D_i with probability at least $1-\delta/K$. After each entity sends its n-bit hypothesis to the center, the center can compute the final hypothesis h which, with probability at most $1-K(\delta/K)=1-\delta$ will have error rate at most ϵ on D.

While the above algorithm achieves a communication complexity growing only linearly in n when the number of entities K is constant, it can be wasteful in terms of both communication and label complexity when K is large, especially for $K\geq\Omega(\log n)$. We can improve upon this first baseline algorithm in such cases by a different fully supervised algorithm that takes advantage of the following general transformation. The idea is to convert any algorithm that learns C in the mistake bound model (with mistake bound M) in the non-distributed setting into a distributed learning algorithm with label complexity evenly distributed among all entities. The transformation is valid for both supervised and semi-supervised algorithms provided we keep track of the communication used.

The transformation is as follows. Since each entity will essentially be running a shadow copy of the same algorithm, they first must communicate to agree on a starting state which should include an initial hypothesis possibly in addition to other information. Often in the fully supervised setting the initial state is predetermined and no communication will be necessary in this step, but in the semi-supervised setting this step may be used to share information about unlabeled data. Since we do not have access to the joint mixture D according to which we are measuring error, we will simulate D by drawing one example at a time from each distribution D_i and repeating the sequence of K independent draws as necessary. When it is entity i's turn, it will draw one example from D_i, attempt to classify it with the current hypothesis, and communicate an update to the hypothesis if a misclassification occurs. This process continues for each entity in turn until some hypothesis correctly classifies $(1/\epsilon)\log(M/\delta)$ consecutive examples, at which point the algorithm outputs the current hypothesis.

Theorem 15. *Given an algorithm A for learning C with mistake bound M, let r be the number of bits of communication required to initialize a common state among all entities and let s be the number of bits required to communicate each hypothesis update. Then there exists an algorithm A' that learns C in the distributed learning model with communication complexity $O(r + Ms)$ and label complexity $O((M/K\epsilon)\log(M/\delta))$ per entity.*

Proof. The communication complexity result is immediate for any A' that uses one state initialization step, one hypothesis update for each mistake, and no additional communication. Since any mistake bound algorithm can be converted into a conservative mistake bound algorithm (one that only updates its hypothesis on a mistake) without a change in the mistake bound, we only need to ensure that no communication is used other than initialization and hypothesis updates, which is clear from the construction.

Since at most M mistakes can be made and there are at most $(1/\epsilon)\log(M/\delta)$ labels seen between mistakes, the overall label complexity is $O((M/\epsilon)\log(M/\delta))$. Since the labeled examples are evenly distributed among the K entities, the label complexity is $O((M/K\epsilon)\log(M/\delta))$ labeled examples per entity.

It only remains to show that the final hypothesis meets the accuracy and confidence requirements. We will only fail to meet the guarantee if one of the at most M hypotheses we consider fools us by correctly classifying $m = (1/\epsilon)\log(M/\delta)$ examples consecutively when its error rate is greater than ϵ. For a fixed hypothesis h with $\mathsf{err}(h) > \epsilon$, let $p_i = 1 - \mathsf{err}_i(h)$ be the probability of correctly classifying an example drawn from D_i. By the definition of D as the uniform mixture of all D_i, we have $K^{-1}\sum_{i=1}^{K} p_i = 1 - \mathsf{err}(h) < 1 - \epsilon$. Since all examples are drawn independently, the probability that we are fooled by h is at most

$$\prod_{i=1}^{K} p_i^{m/K} = \left(\prod_{i=1}^{K} p_i\right)^{m/K} \leq \left(\frac{1}{K}\sum_{i=1}^{K} p_i\right)^{m} < (1-\epsilon)^m \leq \frac{\delta}{M}$$

where the first inequality uses the fact that the geometric mean of nonnegative real numbers is always at most the arithmetic mean. A union bound over the at most M hypotheses tested completes the proof. □

By using this approach to transform the well-known algorithm for learning disjunctions with mistake bound $M = n$, we can achieve a label complexity of $(n/K\epsilon)\log(n/\delta)$ per entity. The initial disjunction is fixed to include all variables and need not be communicated, and on each mistake $\log n$ bits must be sent to indicate which variable to remove from the disjunction.

3.6.2 Absence of Non-indicators

We now give two algorithms to learn two-sided disjunctions with no non-indicators in the distributed learning model. The first algorithm is analogous to the first baseline algorithm in that each entity learns and shares a hypothesis of its own and the hypotheses are combined at the end. In particular each entity starts by drawing enough unlabeled data to construct an accurate distribution graph and then runs the following mistake bound algorithm. Start with all variables labeled as negative indicators. If a mistake is made, set all variables in the corresponding connected component to positive indicators. Entity i can make at most T_i mistakes, where T_i is the number of connected components in the distribution graph of

D_i. If each entity i runs this mistake bound algorithm until it has a hypothesis that with probability at least $1 - \delta/K$ has ϵ error on D_i, the K n-bit hypotheses can be shared and combined by taking the union of all positive indicators.

The second algorithm for this setting takes advantage of Theorem 15 by transforming the above mistake bound algorithm into its distributed version. Each entity begins by drawing enough unlabeled data to construct an accurate distribution graph and then shares the connected component structure of its graph ($O(n \log n)$ bits) with the other entities. The distributed mistake bound algorithm proceeds using the combined distribution graph, and requires indicating which connected component to update for each mistake. Letting T be the number of connected components in the distribution graph of D, this algorithm makes at most T mistakes and uses $\log T$ bits of communication per mistake.

3.6.3 Main Result

With the inclusion of k non-indicators, the distributed algorithm becomes more involved. The simplest solution is for each entity to share its entire distribution graph and to to use the mistake bound algorithm behind Algorithm 3 to achieve a label complexity of $O(\frac{k+T}{K\epsilon} \log \frac{k+T}{\delta})$ with communication complexity $O(Kn^2)$ bits. However, we can improve the communication complexity to $O(Kn \log n)$ bits in the following way.

Each player sends out the component information of G_i', where G_i' is the graph obtained by removing all potential indicators from G_i. Each player also sends out a table that lists for each vertex, whether it is connected by an edge to a potential indicator in G_i. When given the vertex v, the player can build the graph $G' = \cup_i G_i'$, and check in the component of G' that v falls into, whether there is a vertex that is connected by an edge to some potential indicator.

The players can identify the desired potential indicators using this method for the following two facts. 1: G' is the graph obtained by removing all potential indicator from G. 2: A potential indicator is connected in G to v by a path without any other potential indicators in the middle if and only if in the component of G' that v falls into, there is a vertex connected by an edge to this potential indicator.

Since each component can be indexed using $O(\log n)$ bits, describing for each vertex which component it falls into takes $O(n \log n)$ bits. Similarly, each potential can be indexed using $O(\log n)$ bits, and describing for each vertex which potential indicator it is directly connected to takes $O(n \log n)$ bits.

Note: In distributed learning two-sided disjunctions, when a non-indicator is found and removed, the players do not need to send out further information to update G' and the tables. This is because the non-indicator must be a potential indicator by the hitting assumption and its removal does not change G', and because the tables can be updated locally. Also, when the example drawn is from a component that has no potential indicator, no further communication is needed to update G' and the tables. The players know by the hitting assumption that all variables in the component are indicators of the same type as the example, so further examples from this component can be simply labeled without resorting to G' and the tables.

3.7 **Bounding** T **by** $\log |C_{D,\chi}(\epsilon)|$

The following lemma bounds the number of connected components in the commonality graph by the number of compatible hypotheses. For notational definitions, refer to Section 2 in the main body of the paper.

Lemma 8. *Let G be the graph that results from removing all non-indicators from $G_{\text{com}}(U)$, and suppose G is divided into T connected components. If $m_u \geq \frac{2n^2}{\epsilon} \log \frac{n}{\delta}$, then $T \leq \log_2 |C_{D,\chi}(\epsilon)|$ with probability at least $1 - \delta$.*

Proof. Since G has no non-indicators, a hypothesis is compatible with U if and only if every component is made entirely of indicators of the same type. There are two possible choices for each component, so the number of fully compatible hypotheses is $|C_{U,\chi}(0)| = 2^T$.

To complete the proof, it is sufficient to show that $C_{U,\chi}(0) \subseteq C_{D,\chi}(\epsilon)$. Since any hypothesis in $C_{U,\chi}(0)$ is compatible with any example containing variables from only one component, we only need to show that there is at most ϵ probability mass of examples that contain variables from multiple components. All such examples correspond to edges that are absent from $G_{\text{com}}(U)$, so we only need to show that $G_{\text{com}}(U)$ was constructed with enough examples so that nearly all significant edges appear in the graph.

To see this, fix any pair of variables x_i, x_j. If $\Pr_{x\sim D}[x_i = 1 \wedge x_j = 1] < \epsilon/n^2$, we can ignore this pair since all such pairs together constitute a probability mass strictly less than ϵ. Now suppose $\Pr_{x\sim D}[x_i = 1 \wedge x_j = 1] \geq \epsilon/n^2$. The probability that x_i and x_j do not appear together in any of the examples in U is at most $\left(1 - \frac{\epsilon}{n^2}\right)^{m_u}$, so if $m_u \geq \frac{n^2}{\epsilon} \log \frac{n^2}{\delta}$ then this failure probability is at most δ/n^2. By a union bound over all such pairs, with probability at least $1 - \delta$ all corresponding edges appear in $G_{\text{com}}(U)$, and the probability mass of examples containing variables from multiple components is at most ϵ. This means that every fully compatible hypothesis has unlabeled error at most ϵ, so we have $T = \log_2 |C_{U,\chi}(0)| \leq \log_2 |C_{D,\chi}(\epsilon)|$. □

3.8 Finding a Consistent Compatible Hypothesis is NP-hard

The following theorem formalizes the computational difficulty of finding a fully consistent and compatible two-sided disjunction in the semi-supervised setting.

Theorem 16. *Given data sets L and U, finding a hypothesis $h \in C$ that is both consistent with L and compatible with U is* ***NP****-hard.*

Proof sketch. The proof is by reduction from 3-SAT. Given a 3-SAT instance φ on variables $x_1, \ldots, x_n$ we produce the following data sets L and U containing examples on the $4n$ variables $x_1^+, x_1^-, \bar{x}_1^+, \bar{x}_1^-, \ldots, x_n^+, x_n^-, \bar{x}_n^+, \bar{x}_n^-$. The labeled set L contains examples of the form $(\{x_i^+, \bar{x}_i^+\}, +1)$ and $(\{x_i^-, \bar{x}_i^-\}, -1)$ for $1 \leq i \leq n$. In addition, for each clause in φ of the form $(\ell_i \vee \ell_j \vee \ell_k)$ where ℓ_i, ℓ_j, ℓ_k can each be positive or negative literals, L contains the example $(\{\ell_i^+, \ell_j^+, \ell_k^+\}, +1)$. The unlabeled set U contains examples of the form $\{x_i^+, x_i^-\}$ and $\{\bar{x}_i^+, \bar{x}_i^-\}$ for $1 \leq i \leq n$. The labelings that are consistent and compatible with all the non-clause examples correspond precisely to assignments of $x_1, \ldots, x_n$, and the clauses are compatible with a given hypothesis only if they are satisfied in the underlying assignment. The set of positive indicators of any hypothesis $h = (h_+, h_-) \in C$ that is both consistent with L and compatible with U corresponds to a truth assignment to $x_1, \ldots, x_n$ that satisfies φ, therefore finding such a hypothesis is **NP**-hard. □

3.9 Random Classification Noise

Here we consider the problem of learning two-sided disjunctions under random classification noise, where the label of each example is flipped with probability $0 \leq \alpha < 1/2$ independently. Our goal is to extend our algorithms to this setting so that they still successfully output a low error hypothesis without significant increase in sample complexity.

More specifically, we have a distribution $D_{X,Y}$ over labeled examples $(X, \ell(X))$, and the Bayes decision rule is a two-sided disjunction $h^* \in C$, which we also refer to as the target concept. We have $\chi(h^*, D) = 1$ where D is the margin of $D_{X,Y}$ over X, and

$$\Pr[\ell(X) = -h^*(x) | X = x] = \alpha.$$

For a hypothesis h, let $\mathsf{err}_D(h)$ denote its error over the distribution $D_{X,Y}$, i.e.

$$\mathsf{err}_D(h) = \Pr_{(x,\ell(x)) \sim D_{X,Y}} [h(x) \neq \ell(x)].$$

Let $\mathsf{err}_L(h)$ denote its empirical error on the noisy labeled examples L, i.e.

$$\mathsf{err}_L(h) = \frac{1}{|L|} \sum_{(x,\ell(x)) \in L} I[h(x) \neq \ell(x)].$$

For convenience, we define the distance between h and h^* to be

$$d(h, h^*) = \Pr[h(x) \neq h^*(x)].$$

We aim to find a hypothesis h with error

$$\mathsf{err}_D(h) \leq \mathsf{err}_D(h^*) + \epsilon = \alpha + \epsilon.$$

Note that it is sufficient to have $d(h, h^*) \leq \epsilon$, since by triangle inequality

$$\begin{aligned} \mathsf{err}_D(h) &= \Pr[h(x) \neq \ell(x)] \\ &\leq \Pr[h(x) \neq h^*(x)] + \Pr[h^*(x) \neq \ell(x)] \\ &\leq d(h, h^*) + \mathsf{err}_D(h^*) = d(h, h^*) + \alpha. \end{aligned}$$

In the following two subsections, we show how to extend the algorithms (Algorithm 2 and 3) for semi-supervised and active learning respectively. In the last subsection, we include the extension of Algorithm 1. Note that due to the noise in the labeled examples, we cannot hope to find a consistent and compatible hypothesis. The extension of Algorithm 1 only outputs a hypothesis that has low error. However, it achieves better sample complexity bound than the extension of Algorithm 2.

3.9.1 Semi-supervised Learning

In the noise-free setting, we build a hypothesis based on the commonality graph, and then check and update the hypothesis until it has low error rate. The key idea in the noisy setting is that we label variables (i.e. identify variables as positive/negative potential indicators) by majority labels of sufficiently many examples containing the variables, and we use a sufficiently large set of labeled examples each time we check the hypothesis. A brief description is provided below, while the details are provided in Algorithm 5 and Theorem 17.

First, we build the commonality graph and the potential indicator sets. We only label a variable if it is present in at least $\tilde{O}\left(\frac{1}{(1-2\alpha)^2}\right)$ examples so that we can use majority label of the examples to correctly decide its type. We call such variables *significant*. If we draw $\tilde{O}\left(\frac{1}{\epsilon_0(1-2\alpha)^2}\right)$ examples, then with high probability each non-indicator is significant and thus labeled.

Then we build a hypothesis and draw a set of labeled examples to check it. If the empirical error is small, we output the hypothesis since it is guaranteed to have small error. Otherwise, either a component without any labeled variables in it or a non-indicator causes large error. In the first case, we need sufficiently many examples fall in the component so that we can use majority voting to decide the type of the variables in it. This requires $\tilde{O}(\frac{T}{\epsilon(1-2\alpha)^2})$ examples where T is the number of connected components in the graph that results from removing all non-indicators from the commonality graph. In the second case, to reveal the non-indicator, we must be able to distinguish between an error rate of α (the error rate caused by noise) and $\alpha + \Theta(\epsilon(1-2\alpha)/k)$ (the error rate caused by noise and the non-indicator leading to large error). This requires $\tilde{O}(\frac{k^2}{\epsilon^2(1-2\alpha)^2})$ labeled examples for each significant variable. Therefore, we draw $\tilde{O}\left(\frac{1}{(1-2\alpha)^2}\left[\frac{k^3}{\epsilon^3}+\frac{T}{\epsilon}\right]\right)$ examples at each check, so that either a component that previously contained no labeled variables is labeled, or

Algorithm 5 Learning a Low-error Hypothesis for Two-Sided Disjunctions under Random Classification Noise

```
Input: data sets $U$ and $L$, parameters $\epsilon$, $\delta$, $k$, $\epsilon_0$, $T$
Set $G = G_{\text{com}}(U), V_+^0 = V_-^0 = \emptyset$
Set $L' = \text{sample}\left(\frac{100}{\epsilon_0(1-2\alpha)^2}\log^2\frac{n}{\delta}, L\right)$ and $L = L \setminus L'$
Set $size(v) = |\{x \in L' : x \ni v\}|, \forall v \in V$
Set $SIG = \{v \in V : size(v) > \frac{10}{(1-2\alpha)^2}\log\frac{n}{\delta}\}$
for each $v \in SIG$ do
    Set $l = \text{sign}(\sum_{x \ni v} \ell(x))$ and $V_l^0 = V_l^0 \cup \{v\}$
end for
Set $V_+ = V_+^0, V_- = V_-^0$ and $h = h_{G,V_+\cup V_-}$
$S = \text{sample}\left(\frac{100}{(1-2\alpha)^2}\left[\frac{k^3}{\epsilon^3} + \frac{T}{\epsilon}\right]\log^2\frac{n}{\delta}, L\right)$, $L = L \setminus S$
while $L \neq \emptyset$ and $\mathsf{err}_S(h) > \alpha + \frac{1-2\alpha}{2}\epsilon$ do
    Let $\mathcal{R}$ denote the set of the components in $G$ that have no labeled variables
    Let $S(R) = \{x \in S : x \text{ falls into } R\}, \forall R \in \mathcal{R}$
    Let $S(v) = \{x \in S : \text{nn}_{G,V_+\cup V_-}(x) = v\}, \forall v \in V_+ \cup V_-$
    if $\exists R \in \mathcal{R}$ such that $|S(R)| \geq \frac{10}{(1-2\alpha)^2}\log\frac{n}{\delta}$ then
        Set $l = \text{sign}(\sum_{x \in S(R)} \ell(x))$ and $V_l = V_l \cup R$
    end if
    if $\exists v \in V_+^0 \cup V_-^0$ such that $|S(v)| \geq \frac{10k^2}{\epsilon^2(1-2\alpha)^2}\log\frac{n}{\delta}$ and $\mathsf{err}_{S(v)}(h) \geq \alpha + (1-2\alpha)\frac{\epsilon}{16k}$
    then
        Set $G = G \setminus \{v\}$
    end if
    Set $S = \text{sample}\left(\frac{100}{(1-2\alpha)^2}\left[\frac{k^3}{\epsilon^3} + \frac{T}{\epsilon}\right]\log^2\frac{n}{\delta}, L\right)$,
    Set $L = L \setminus S$ and $h = h_{G,V_+\cup V_-}$
end while
Output: the hypothesis $h$
```

a non-indicator is revealed. After at most $(k+T)$ updates, we are guaranteed to have a hypothesis with small error.

Theorem 17. *For any distribution $D_{X,Y}$ over $\{0,1\}^n \times \{-1,1\}$ and target concept $h^* \in C$ in the random classification noise model such that h^* has at most k non-indicators, and the minimum non-indicator probability is ϵ_0, if $m_u = O(\frac{n^2}{\epsilon}\log\frac{n}{\delta})$ and*

$$m_l = O\bigg(\frac{k + \log|C_{D,\chi}(\epsilon)|}{(1-2\alpha)^2}\log^2\frac{n}{\delta} \left[\frac{1}{\epsilon_0} + \frac{k^3}{\epsilon^3} + \frac{\log|C_{D,\chi}(\epsilon)|}{\epsilon}\right]\bigg)$$

then with probability at least $1-\delta$, Algorithm 5 outputs a hypothesis h in polynomial time such that $\mathsf{err}_D(h) \leq \alpha + \epsilon$.

Proof. **Generalization Error:** Suppose we check the hypothesis at most $k+T$ times (proved later), where T is the number of connected components in the graph that results from removing all non-indicators from $G_{\text{com}}(U)$. Then when

$$|S| \geq \frac{100}{(1-2\alpha)^2\epsilon^2} \log \frac{k+T}{\delta}$$

w.h.p. all hypotheses h with $d(h,h^*) > \epsilon$ will have empirical error on S larger than $\alpha + \frac{1-2\alpha}{2}\epsilon$. So when the algorithm stops the hypothesis satisfies $d(h,h^*) \leq \epsilon$ and thus $\mathsf{err}_D(h) \leq \alpha + \epsilon$.

Bounding the Number of Updates: We now show that the hypothesis is indeed updated at most $(k+T)$ times. We begin by proving that when

$$|L'| \geq \frac{100}{\epsilon_0(1-2\alpha)^2} \log^2 \frac{n}{\delta}$$

w.h.p. every non-indicator is labeled and every labeled indicator gets the correct label, so that the hypothesis is updated correctly when its error is large. First, by Chernoff and union bounds, the probability that there exists a non-indicator that appears in less than $\frac{10}{(1-2\alpha)^2}\log\frac{n}{\delta}$ examples is bounded by $k\exp\{O(\epsilon_0|L'|)\} \leq O(\delta)$. So w.h.p. every non-indicator appears in enough examples and thus is labeled. Second, the type of an indicator is decided by the majority label of $O(\frac{1}{(1-2\alpha)^2}\log\frac{n}{\delta})$ examples. By Hoeffding and union bounds, w.h.p. the types of all indicators appearing in enough examples are decided correctly.

We now prove that when the error of the hypothesis is large, we can make progress by either labeling a previously unlabeled component or identifying a non-indicator, and thus it is updated at most $(k+T)$ times. When

$$|S| > \frac{100}{(1-2\alpha)^2\epsilon^2} \log \frac{k+T}{\delta},$$

if $\mathsf{err}_S(h) > \alpha + \frac{1-2\alpha}{2}\epsilon$, then w.h.p. $d(h,h^*) \geq \epsilon/4$. For each $v \in V_+ \cup V_-$, let $X(v)$ denote those examples whose nearest labeled variable is v, i.e.

$$X(v) = \{x \in X : \mathrm{nn}_{G,V_+\cup V_-}(x) = v\}.$$

For each $R \in \mathcal{R}$, let $X(R)$ denote those examples fall into the component R, i.e.

$$X(R) = \{x \in X : x \text{ falls into } R\}.$$

Note that for any indicator $v \in V_+ \cup V_-$, its type is correctly decided, so the hypothesis makes no mistake on $X(v)$. Since $|\mathcal{R}| \leq T$ and the number of non-indicators is bounded by k, either there is a component R' such that

$$\Pr[h(x) \neq h^*(x) \wedge x \in X(R')] > \epsilon/(8T)$$

or there is a non-indicator v' such that

$$\Pr[h(x) \neq h^*(x) \wedge x \in X(v')] > \epsilon/(8k).$$

In the first case, w.h.p. there are more than $\frac{10}{(1-2\alpha)^2} \log \frac{n}{\delta}$ examples in $S(R')$ when

$$|S| \geq \frac{100T}{\epsilon(1-2\alpha)^2} \log^2 \frac{n}{\delta}.$$

These examples are sufficient to decide the type of the indicators in the component correctly. Also, w.h.p. for any $R \in \mathcal{R}$ with more than $\frac{10}{(1-2\alpha)^2} \log \frac{n}{\delta}$ examples in $S(R)$, the type of the indicators in the component are correctly decided. This means that we correctly label at least one component not labeled previously. This type of updates happen at most T times.

In the second case, we have

$$\Pr[h(x) \neq h^*(x) | x \in X(v')] > \epsilon/(8k)$$

and thus

$$\Pr[h(x) \neq \ell(x) | x \in X(v')] > \alpha + (1-2\alpha)\epsilon/(8k).$$

When

$$|S| \geq \frac{100k^3}{\epsilon^3(1-2\alpha)^2} \log^2 \frac{n}{\delta}$$

w.h.p. in $S(v')$ there are more than $\frac{10k^2}{\epsilon^2(1-2\alpha)^2} \log \frac{n}{\delta}$ examples and we have

$$\mathsf{err}_{S(v')}(h) > \alpha + (1-2\alpha)\epsilon/(16k).$$

Also, for any indicator $v \in V_+ \cup V_-$,

$$\Pr[h(x) \neq \ell(x) | x \in X(v)] = \alpha.$$

Then w.h.p. we have

$$\mathsf{err}_{S(v)}(h) < \alpha + (1-2\alpha)\epsilon/(16k)$$

for any indicator v such that $|S(v)| \geq \frac{10k^2}{\epsilon^2(1-2\alpha)^2} \log \frac{n}{\delta}$. This means that we can correctly identify a non-indicator. This type of updates happen at most k times. Hence, we update the hypothesis at most $(k+T)$ times.

Sample Complexity and Running Time: When building the commonality graph, we need

$$|L'| = O\left(\frac{1}{\epsilon_0(1-2\alpha)^2} \log^2 \frac{n}{\delta}\right).$$

Each time we check the hypothesis, we need

$$|S| = O\left(\frac{1}{(1-2\alpha)^2}\left[\frac{k^3}{\epsilon^3} + \frac{T}{\epsilon}\right] \log^2 \frac{n}{\delta}\right).$$

The number of labeled examples then follows from bounding the number of checks by $(k+T)$ and bounding T by $\log |C_{D,\mathcal{X}}(\epsilon)|$. The algorithm runs in polynomial time since building the commonality graph and checking the hypothesis take polynomial time. □

3.9.2 Active Learning

There are two main changes in extending our algorithm to the noisy setting. First, instead of simply determining the type of an indicator with one example, we need to check many examples that contain it. A majority vote will correctly identify the type of the indicator. Second, instead of doing binary search over nodes in the path that connect positive and negative examples, we need to search over edges. This is because with noise, an indicator may appear both in negative and positive examples similar to a non-indicator, so it is not straightforward to identify a non-indicator merely by the types of examples it appears in. On the other hand, an edge that contains an indicator will have its true label match that of the indicator, so we can reliably determine the type of an edge if there are enough examples that contain both variables, and then identify a vertex in edges of two different types to be a non-indicator. More specifically, Subroutine 6 can be used to decide the type of a pair of variables (when $u \neq v$) or that of one variable (when $u = v$), which is a building block for the extension of the active learning algorithm to the noisy setting. A brief description of the extension is provided below, while the details are provided in Algorithm 7 and Theorem 18.

First, we build the commonality graph and an initial hypothesis. Let F be the set of all examples that contain some pair of variables appearing together in fewer than $O(\frac{1}{(1-2\alpha)^2} \log \frac{n}{\delta})$

Subroutine 6 DecideType(U, u, v)

```
Input: unlabeled data $U$, variables $u$ and $v$
Set $S = \text{sample}(\frac{100}{(1-2\alpha)^2}\log\frac{n}{\delta}, \{x \in U : \{u,v\} \subseteq x\})$
Set $t = \text{sign}(\sum_{x\in S}\ell(x))$
Output: $\{(u,t),(v,t)\}$
```

examples. We only use the unlabeled data $U \setminus F$ to construct the commonality graph, so that every edge corresponds to a pair of variables whose indicator type can be decided. Then we pick a variable in each component in the graph and decide its type. This requires $\tilde{O}(\frac{T}{(1-2\alpha)^2})$ queries, where T is the number of connected components in the graph that results from removing all non-indicators from the commonality graph $G_{\text{com}}(U \setminus F)$. A hypothesis is then constructed which labels an input example by the type of the nearest labeled variable.

Second, we check and update the hypothesis on a set of examples. We randomly sample a set S of $\tilde{O}(\frac{1}{(1-2\alpha)^2\epsilon^2})$ examples from $U \setminus F$, and compute $\mathsf{err}_S(h)$. If $\mathsf{err}_S(h)$ is at most $\alpha + \frac{1-2\alpha}{2}\epsilon$, we output the hypothesis since it has small error. Otherwise, it can be shown that on $\Omega(\epsilon)$ fraction of examples in S the hypothesis has different labels from the target concept h^*. This fact can be used to identify a non-indicator. We randomly sample $\tilde{O}(\frac{1}{\epsilon})$ examples from S, and for each example x, pick $\min(k+1, |x|_1)$ variables and decide their types, where $|x|_1$ is the number of variables appearing in x. This ensures that we will eventually pick an indicator in an example x such that $h(x) \neq h^*(x)$. Then we find a path connecting a positive indicator and a negative indicator, and thus can identify a non-indicator by binary search on the edges along the path. Therefore, after at most k updates, we are guaranteed to have a hypothesis with small error.

Theorem 18. *For any distribution $D_{X,Y}$ over $\{0,1\}^n \times \{-1,1\}$ and target concept $h^* \in C$ in the random classification noise model such that h^* has at most k non-indicators, if $|U| = O\left(\frac{n^2}{\epsilon(1-2\alpha)^2}\log^2\frac{n}{\delta}\right)$, after at most*

$$m_q = O\left(\frac{1}{(1-2\alpha)^2}\left[\log|C_{D,X}(\epsilon)| + \frac{k^2}{\epsilon} + \frac{k}{\epsilon^2}\right]\log^2\frac{n}{\delta}\right)$$

label queries, with probability at least $1-\delta$, Algorithm 7 outputs a hypothesis h in polynomial time such that $\mathsf{err}_D(h) \leq \alpha + \epsilon$.

Algorithm 7 Actively Learning Two-Sided Disjunctions under Random Classification Noise

```
Input: unlabeled data $U$, parameters $\alpha, \epsilon, \delta, k$
Set $U(u,v) = |\{x \in U : \{u,v\} \subseteq x\}|, \forall u,v \in V$.
Set $F = \{x \in U : \exists u,v \in x,\, U(u,v) < \frac{100}{(1-2\alpha)^2} \log \frac{n}{\delta}\}$
Set $U' = U \setminus F$, $G = G_{\text{com}}(U')$, and $L = \emptyset$
for each connected component $R$ of $G$ do
    Set $L = L \cup \text{DecideType}(U, v, v)$ for any $v \in R$
end for
Set $h = h_{G,L}$ and $S = \text{sample}(\frac{10}{(1-2\alpha)^2\epsilon^2} \log \frac{k}{\delta}, U')$
while $\mathsf{err}_S(h) > \alpha + \frac{1-2\alpha}{2}\epsilon$ do
    for $i = 1$ to $\frac{100}{\epsilon} \log \frac{k}{\delta}$ do
        Set $x = \text{sample}(1, S)$
        for each of $\min(k+1, |x|_1)$ variables $v \in x$ do
            Set $L = L \cup \text{DecideType}(U, v, v)$
        end for
        if $\exists (u,1), (v,-1) \in L$ such that $u \leftrightarrow_G v$ then
            Set $v = \text{BinarySearch}_{G,L}(x)$
            Set $G = G \setminus \{v\}$
            for each new component $R$ of $G$ do
                Set $L = L \cup \text{DecideType}(U, v, v)$ for $v \in R$
            end for
            Set $h = h_{G,L}$
            break
        end if
    end for
    Set $S = \text{sample}(\frac{10}{(1-2\alpha)^2\epsilon^2} \log \frac{k}{\delta},\ U')$
end while
Output: the hypothesis $h$
```

Proof. **Generalization Error:** Assuming the hypothesis is updated at most k times (proved later), we bound the probability that the output hypothesis h has $d(h, h^*) \leq \epsilon$. We begin by showing that the ignored examples F have small probability mass. When U is sufficiently large, w.h.p. all pairs of variables that appear together with probability at least $\frac{\epsilon}{8n^2}$ will appear in sufficiently many examples in U. Assuming this is true, we have

$$\Pr[x \in F] \leq \epsilon/8.$$

This means when $d(h, h^*) > \epsilon$,

$$\Pr[h(x) \neq h^*(x) | x \in X \setminus F] > 3\epsilon/4$$

and

$$\Pr[h(x) \neq \ell(x) | x \in X \setminus F] > \alpha + (1-2\alpha)\frac{3\epsilon}{4}.$$

Then we have

$$\Pr\left[\mathsf{err}_S(h) \le \alpha + \frac{1-2\alpha}{2}\epsilon\right] \le \frac{\delta}{8k}.$$

Union bounding over the k updates, we have that w.h.p. the hypothesis h output has $d(h, h^*) \le \epsilon$, which leads to $\mathsf{err}_D(h) \le \alpha + \epsilon$.

Correctness of Subroutine 6: Here we show that w.h.p. the majority voting method always decides correctly the type of the indicators, so that we build and update the hypothesis correctly. Fix a pair of variables (u, v) containing at least one indicator. Let $B_{u,v}$ denote the event that (u, v) appear in at least $\frac{100}{(1-2\alpha)^2} \log \frac{n}{\delta}$ examples in U but the algorithm fails to decide the type. This happens when the labels of more than half of the examples queried are flipped. We have by Hoeffding bound

$$\Pr[B_{u,v}] \le \exp\left\{-2(1-2\alpha)^2 \frac{100}{(1-2\alpha)^2} \log \frac{n}{\delta}\right\} \le \frac{\delta}{4n^2}$$

and thus $\Pr[\cup_{u,v} B_{u,v}] \le \frac{\delta}{4}$.

Queries per Stage: To build the hypothesis, we decide the type of one variable for each connected component of G. The number of components is bounded by T, so here we need $O(\frac{T}{(1-2\alpha)^2} \log \frac{n}{\delta})$ queries.

We now show that by using sufficient many queries at each check, we make sure that when the hypothesis has large error a non-indicator is identified, so that the hypothesis is updated at most k times. If $d(h, h^*) \le \epsilon/4$, then

$$\Pr[h(x) \ne h^*(x) | x \in X \setminus F] \le \epsilon/3$$

and thus

$$\Pr[h(x) \ne \ell^*(x) | x \in X \setminus F] \le \alpha + (1-2\alpha)\frac{\epsilon}{3}.$$

Then w.h.p. when $|S| = O(\frac{1}{(1-2\alpha)^2\epsilon^2} \log \frac{k}{\delta})$,

$$\mathsf{err}_S(h) \le \alpha + \frac{1-2\alpha}{2}\epsilon.$$

Therefore, if $\mathsf{err}_S(h) > \alpha + \frac{1-2\alpha}{2}\epsilon$, we have $d(h, h^*) \ge \epsilon/4$. This means on at least $\epsilon/16$ fraction of the examples in S we have $h(x) \ne h^*(x)$. By sampling $\frac{100}{\epsilon} \log \frac{k}{\delta}$ times from S and then picking $\min(k+1, |x|_1)$ variables in the sampled x, w.h.p. we will eventually pick such an example, and pick at least one indicator in it, whose type is different from the nearest indicator. Then we find a path connecting positive and negative indicators, and discover a non-indicator by binary search. So we need

$$O\left(\frac{1}{(1-2\alpha)^2\epsilon^2}\log\frac{k}{\delta}+\frac{k}{(1-2\alpha)^2\epsilon}\log\frac{k}{\delta}\right)$$

queries each time we check and update the hypothesis.

Query Complexity and Running Time: Since when the hypothesis has large error a non-indicator is identified, it is checked and updated at most k times. Then the number of queries follows by bounding T by $\log|C_{D,\mathcal{X}}(\epsilon)|$. Notice that T is the number of connected components in $G_{\mathrm{com}}(U\setminus F)$ (instead of $G_{\mathrm{com}}(U)$) after removing the non-indicators. However, when U is sufficiently large, w.h.p. the probability mass of F is at most $\epsilon/8$, i.e. all significant edges appear in the graph, so we still have $T \leq \log|C_{D,\mathcal{X}}(\epsilon)|$. Building, checking and updating the hypothesis all take polynomial time, so the algorithm runs in polynomial time. □

3.10 Discussion

One drawback of our semi-supervised algorithms is that their dependence on the minimum non-indicator probability restricts the class of distributions under which they can be used for learning. Additionally, the class of target concepts for which Algorithm 2 can efficiently learn a consistent and compatible hypothesis is restricted, and the reduction given in Section 3.8 proves that some such restriction is necessary as the general problem is **NP**-hard. One surprising result of our work is that both restrictions can be lifted entirely in the active learning setting while improving label complexity at the same time. The ability to adaptively query the labels of examples allows us to execute a strategy for identifying non-indicators that would require too many labeled examples in the semi-supervised setting. While this represents the first known example of how active learning can be used to avert computational difficulties present in semi-supervised learning, it would be worthwhile to give more such examples and to understand more generally when active learning provides this type of advantage.

It is important to note that the problem of learning two-sided disjunctions can be viewed as learning under a large-margin assumption. We can represent a two-sided disjunction h as a linear threshold function $h(x) = \mathrm{sign}(w^\mathsf{T} x)$ where $w_i = +1$ for positive indicators, $w_i = -1$ for negative indicators, and $w_i = 0$ for each of the k non-indicators. If h is fully compatible with the distribution D, it is straightforward to show that for every example $x \sim D$, $\frac{|w^\mathsf{T} x|}{\| w \|_\infty \| x \|_1} \geq \frac{1}{k+1}$, which corresponds to an $L_\infty L_1$ margin of $O(1/k)$. This is a different notion of margin than the $L_2 L_2$ margin appearing in the mistake bounds for the Perceptron algorithm [79] and the $L_1 L_\infty$ margin appearing in the bounds for Winnow [63]. One interesting area of future work is to provide generic algorithms with bounds depending on the $L_\infty L_1$ margin.

CHAPTER IV

DISTRIBUTED CLUSTERING

4.1 Introduction

Most classic clustering algorithms are designed for the centralized setting, but in recent years data has become distributed over different locations, such as distributed databases [74, 31], images and videos over networks [65], surveillance [43] and sensor networks [30, 44]. In many of these applications the data is inherently distributed because, as in sensor networks, it is collected at different sites. As a consequence it has become crucial to develop clustering algorithms which are effective in the distributed setting.

Several algorithms for distributed clustering have been proposed and empirically tested. Some of these algorithms [41, 84, 36] are direct adaptations of centralized algorithms which rely on statistics that are easy to compute in a distributed manner. Other algorithms [49, 54] generate summaries of local data and transmit them to a central coordinator which then performs the clustering algorithm. No theoretical guarantees are provided for the clustering quality in these algorithms, and they do not try to minimize the communication cost. Additionally, most of these algorithms assume that the distributed nodes can communicate with all other sites or that there is a central coordinator that communicates with all other sites.

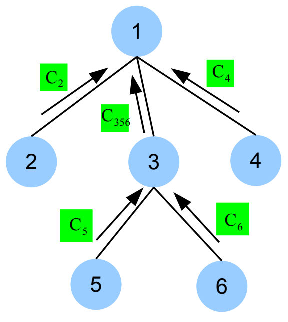
In this chapter, we study the problem of distributed clustering where the data is distributed across nodes whose communication is restricted to the edges of an arbitrary graph. We provide algorithms with small communication cost and provable guarantees on the clustering quality. Our technique for reducing communication in general graphs is based on the construction of a small set of points which act as a proxy for the entire data set.

An ϵ-*coreset* is a weighted set of points whose cost on any set of centers is approximately the cost of the original data on those same centers up to accuracy ϵ. Thus an approximate solution for the coreset is also an approximate solution for the original data. We first propose a distributed algorithm for k-means and k-median, by which each node constructs a local portion of a global coreset. Communicating the approximate cost of a global solution to each node is enough for the local construction, leading to low communication cost overall. The nodes then share the local portions of the coreset, which can be done efficiently in general graphs using a message passing approach. Coresets have previously been studied in the centralized setting ([48, 39]) but have also recently been used for distributed clustering as in [88] and as implied by [39].

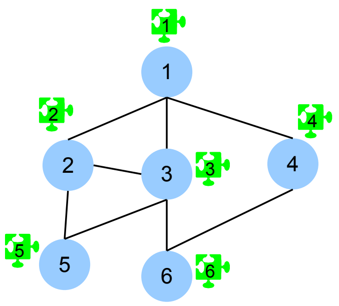
In Section 4.4, we propose a distributed coreset construction algorithm based on local approximate solutions. Each node can construct the local portion of a coreset using only its local data and the total weight of each node's approximation. For ϵ constant, this builds a coreset of size $\tilde{O}(kd + nk)$ for k-median and k-means when the data lies in d dimensions and is distributed over n sites. Building on the distributed coreset construction algorithm, we propose an algorithm for distributed clustering over general connected graphs. The size of the constructed coreset is independent of the communication network topology. This improves over the work of [88] which builds coresets whose size depends on the height of a rooted tree.[1] [40] also merge coresets together using coreset construction, but they do so in a model of parallel computation and ignore communication costs.

Comparison to Other Coreset Algorithms: Since coresets summarize local information they are a natural tool to use when trying to reduce communication complexity. If each node constructs an ϵ-coreset on its local data, then the union of these coresets is clearly an ϵ-coreset for the entire dataset. Unfortunately the size of the coreset using this natural approach increases greatly with the number of nodes. More sophisticated approaches, such as [88] reduce the size of the global coreset by approximating the union of local coresets with another coreset. They assume nodes communicate over a rooted tree, with each node passing its coreset to its parent. Because the approximation factor of the constructed coreset depends on the quality of its component coresets, the accuracy a coreset needs (and thus the overall communication complexity) scales with the height of this tree. Although it is possible

[1]Their result depended on older coreset constructions, with larger size. Throughout this chapter, when we compare to [88] we assume they use the coreset construction technique of [39] to reduce their coreset size and communication cost.



(a) Zhang et al.[88]



(b) Our Construction

Figure 6: **(a)** Each node computes a coreset on the weighted pointset for its own data and its subtrees' coresets. **(b)** Local constant approximation solutions are computed, and the costs of these solutions are used to coordinate the construction of a local portion on each node.

to find a spanning tree in any communication network, when the graph has large diameter every tree has large height. In particular many natural networks such as grid networks have a large diameter ($\Omega(\sqrt{n})$ for grids) which greatly increases the size of coresets which must be communicated across the lower levels of the tree.

We show that it is possible to construct a global coreset with low communication overhead. This is done by distributing the coreset construction procedure rather than combining local coresets. The communication needed to construct this coreset is negligible — just a single value from each data set representing the approximate cost of their local optimal clustering. Since the sampled global ϵ-coreset is the same size as any local ϵ-coreset, this leads to an improvement of the communication cost over the other approaches. See Figure 6 for an illustration.

The constructed coreset is smaller by a factor of n in general graphs, and is independent of the underlying communication topology. This method excels in sparse networks with large diameters, where the previous approaches require coresets that are quadratic in the size of the diameter for k-median and quartic for k-means.

4.2 Related Work

Many empirical algorithms adapt the centralized algorithms to the distributed setting. They generally provide no bound for the clustering quality or the communication cost. For instance, a technique is proposed in [41] to adapt several iterative center-based data clustering algorithms including Lloyd's algorithm for k-means to the distributed setting, where sufficient statistics instead of the raw data are sent to a central coordinator. This approach involves transferring data back and forth in each iteration, and thus the communication cost depends on the number of iterations. Similarly, the communication costs of the distributed clustering algorithms proposed in [36] and [84] depend on the number of iterations. Some other algorithms gather local summaries and then perform global clustering on the summaries. The distributed density-based clustering algorithm in [49] clusters and computes summaries for the local data at each node, and sends the local summaries to a central node where the global clustering is carried out. This algorithm only considers the flat two-tier topology. Some in-network aggregation schemes for computing statistics over distributed data are useful for such distributed clustering algorithms. For example, an algorithm is provided in [30] for approximate duplicate-sensitive aggregates across distributed data sets, such as SUM. An algorithm is proposed in [44] for power-preserving computation of order statistics such as quantile.

As mentioned previously, several coreset construction algorithms have been proposed for k-median, k-means and k-line median clustering [48, 27, 47, 61, 39]. For example, the algorithm in [39] constructs a coreset of size $\tilde{O}(kd/\epsilon^2)$ whose cost approximates that of the original data up to accuracy ϵ with respect to k-median in $\mathcal{B}^d$. All of these algorithms consider coreset construction in the centralized setting, while our construction algorithm is for the distributed setting.

There has also been work attempting to parallelize clustering algorithms. [40] showed that coresets could be constructed in parallel and then merged together. In Scalable `k-means++`, [5] adapted `k-means++` to the parallel setting. In this setting a centralized problem is broken up and distributed to several processors with the aim of reducing computation time. In contrast to the distributed setting, the communication costs are ignored.

There is also related work on clustering on samples [16]. They study the approach of obtaining an approximation solution by clustering only a sample drawn i.i.d. from the data. For k-median, with a sample of size $\tilde{O}(\frac{k}{\epsilon^2})$, they obtain a solution with an average cost bounded by twice that of the optimal average cost plus an error bound ϵ. If we convert it to a multiplicative approximation factor, the factor depends on the optimal average cost. When the optimal average cost is small, the factor becomes large. This happens when there are outlier points far away from all other points. In this case, after normalization, the average cost can be very small, and no good approximation factor is available. The coreset approach provides better guarantees. Additionally, their approach is not applicable to k-means.

4.3 Preliminaries

Let $d(p, q)$ denote the Euclidean distance between any two points $p, q \in \mathcal{B}^d$. The goal of k-means clustering is to find a set of k centers $\mathbf{x} = \{x_1, x_2, \ldots, x_k\}$ which minimize the k-means cost of data set $P \subseteq \mathcal{B}^d$. Here the k-means cost is defined as $\text{cost}(P, \mathbf{x}) = \sum_{p \in P} d(p, \mathbf{x})^2$ where $d(p, \mathbf{x}) = \min_{x \in \mathbf{x}} d(p, x)$. If P is a weighted data set with a weighting function w, then the k-means cost is defined as $\sum_{p \in P} w(p) d(p, \mathbf{x})^2$. Similarly, the k-median cost is defined as $\sum_{p \in P} d(p, \mathbf{x})$. Both k-means and k-median cost functions are known to be **NP**-hard to minimize, so we generally aim at approximation solutions.

In the distributed clustering task, we consider a set of n nodes $V = \{v_i, 1 \leq i \leq n\}$ which communicate on an undirected connected graph $G = (V, E)$ with $m = |E|$ edges. More precisely, an edge $(v_i, v_j) \in E$ indicates that v_i and v_j can communicate with each other. Here we measure the communication cost in number of points transmitted, and assume for simplicity that there is no latency in the communication. On each node v_i, there is a local set of data points P_i, and the global data set is $P = \bigcup_{i=1}^{n} P_i$. The goal is to find a set of k centers $\mathbf{x}$ which optimize $\text{cost}(P, \mathbf{x})$ while keeping the computation efficient and the communication cost as low as possible. Our focus is to reduce the total communication cost while preserving theoretical guarantees for approximating clustering cost.

4.3.1 Coresets

For the distributed clustering task, a natural approach to avoid broadcasting raw data is to generate a local summary of the relevant information. If each site computes a summary for their own data set and then communicates this to a central coordinator, a solution can be computed from a much smaller amount of data, drastically reducing the communication.

In the centralized setting, the idea of summarization with respect to the clustering task is captured by the concept of coresets [48, 39]. A coreset is a set of points, together with a weight for each point, such that the cost of this weighted set approximates the cost of the original data for any set of k centers. The formal definition of coresets is:

Definition 5 (**coreset**). *An ϵ-coreset for a set of points P with respect to a center-based cost function is a set of points S and a set of weights $w : S \to \mathcal{B}$ such that for any set of centers $\mathbf{x}$,*

$$(1-\epsilon)\mathrm{cost}(P,\mathbf{x}) \le \sum_{p\in S} w(p)\mathrm{cost}(p,\mathbf{x}) \le (1+\epsilon)\mathrm{cost}(P,\mathbf{x}).$$

In the centralized setting, many coreset construction algorithms have been proposed for k-median, k-means and some other cost functions. For example, for points in $\mathcal{B}^d$, algorithms in [39] construct coresets of size $t = \tilde{O}(kd/\epsilon^4)$ for k-means and coresets of size $t = \tilde{O}(kd/\epsilon^2)$ for k-median. In the distributed setting, it is natural to ask whether there exists an algorithm that constructs a small coreset for the entire point set but still has low communication cost. Note that the union of coresets for multiple data sets is a coreset for the union of the data sets. The immediate construction of combining the local coresets from each node would produce a global coreset whose size was larger by a factor of n, greatly increasing the communication complexity. We present a distributed algorithm which constructs a global coreset the same size as the centralized construction and only needs a single value[2] communicated to each node. This serves as the basis for our distributed clustering algorithm.

4.4 Distributed Coreset Construction

In this section, we design a distributed coreset construction algorithm for k-means and k-median. Note that the underlying technique can be extended to other additive clustering objectives such as k-line median.

To gain some intuition on the distributed coreset construction algorithm, we briefly review the coreset construction algorithm in [39] in the centralized setting. The coreset is constructed by computing a constant approximation solution for the entire data set, and then

[2]The value that is communicated is the sum of the costs of approximations to the local optimal clustering. This is guaranteed to be no more than a constant factor times larger than the optimal cost.

sampling points proportional to their contributions to the cost of this solution. Intuitively, the points close to the nearest centers can be approximately represented by the nearest centers while points far away cannot be well represented. Thus, points should be sampled with probability proportional to their contributions to the cost.

Directly adapting the algorithm to the distributed setting would require computing a constant approximation solution for the entire data set. We show that a global coreset can be constructed in a distributed fashion by estimating the weight of the entire data set with the sum of local approximations. With this approach, it suffices for nodes to communicate the total costs of their local solutions.

Algorithm 8 Distributed coreset construction

```
1: **Input:** $t, \{P_i, 1 \le i \le n\}$.
2: **Round 1:** on each node $v_i \in V$
3: Compute a constant approximation $B_i$ for $P_i$;
   communicate $\mathrm{cost}(P_i, B_i)$ to all other nodes.
4: **Round 2:** on each node $v_i \in V$
5: Set $t_i = \frac{t\ \mathrm{cost}(P_i,B_i)}{\sum_{j=1}^n \mathrm{cost}(P_j,B_j)}$ and $m_p = \mathrm{cost}(p, B_i), \forall p \in P_i$.
6: Pick a non-uniform random sample $S_i$ of $t_i$ points from $P_i$, where for every $q \in S_i$ and
   $p \in P_i$, we have $q = p$ with probability $m_p / \sum_{q \in P_i} m_q$. Let $w_p = \frac{\sum_i \sum_{q \in P_i} m_q}{t m_p}$ for each
   $p \in S_i$.
7: For $\forall b \in B_i$, let $P_b = \{p \in P_i : d(p,b) = d(p,B_i)\}$, $w_b = |P_b| - \sum_{p \in P_b \cap S_i} w_p$.
8: **Output:** $S_i \cup B_i, \{w_p : p \in S_i \cup B_i\}, 1 \le i \le n$.
```

Theorem 19. *For distributed k-means and k-median clustering on a graph, there exists an algorithm such that with probability at least $1 - \delta$, the union of its output on all nodes is an ϵ-coreset for $P = \bigcup_{i=1}^n P_i$. The size of the coreset is $O(\frac{1}{\epsilon^4}(kd + \log \frac{1}{\delta}) + nk \log \frac{nk}{\delta})$ for k-means, and $O(\frac{1}{\epsilon^2}(kd + \log \frac{1}{\delta}) + nk)$ for k-medians. The total communication cost is $O(mn)$.*

The distributed coreset construction algorithm for k-means and k-median is described in Algorithm 8. Note that there are various constant approximation algorithms available for computing local solutions [53, 62]. In the following subsections, we show that the algorithm constructs a coreset for the global data. We focus on the analysis for k-median, and provide a proof sketch for k-means since the key ideas are similar.

4.4.1 Proof of Theorem 19: k-median

The analysis relies on the definition of the dimension of a function space and a sampling lemma.

Definition 6 ([39]). *Let H be a finite set of functions from a set $\mathbf{X}$ to $\mathcal{B}_{\geq 0}$. Let* $\mathrm{range}(H, \mathbf{x}, r) = \{h \in H : h(\mathbf{x}) \leq r\}$*. The dimension of the function space* $\dim(H, \mathbf{X})$ *is the smallest integer d such that for any $G \subseteq H$, $\left|\{G \cap \mathrm{range}(H, \mathbf{x}, r) : \mathbf{x} \in \mathbf{X}, r \geq 0\}\right| \leq |G|^d$.*

To interpret this, let us investigate a closely related concept class. For points $p \in \mathcal{B}^d$, let $I_{\mathbf{x},r}(p)$ be 1 if $d(\mathbf{x}, p) \leq r$ and -1 otherwise. Equivalently, for a given set of centers $\mathbf{x}$ and radius r, a point is marked as positive if it closer than r to one of the centers.

If the functions h_p in H are indexed by points $p \in P$ such that $h(\mathbf{x}) := \min d(x, p)$, (as in our problem) then the dimension $\dim(H, \mathbf{X})$ is closely related to the number of ways P can be split using concepts from I. In particular we have $\dim(H, \mathbf{X}) = \Theta(\text{VC-Dim}(I))$.

Lemma 9. *Fix a set H of functions $h_p : \mathbf{X} \to \mathcal{B}_{\geq 0}$, a set of weights $m_p \in \mathcal{B}_{>0}$ for $p \in P$. Let S be a sample drawn i.i.d. from P where each p is sampled with probability $\frac{m_p}{\sum_{q \in P} m_q}$, and let $w_p = \frac{\sum_{q \in P} m_q}{m_p |S|}$ for $p \in S$. If for a sufficiently large c, $|S| \geq \frac{c}{\epsilon^2}\left(\dim(H, \mathbf{X}) + \log \frac{1}{\delta}\right)$ then with probability at least $1 - \delta, \forall \mathbf{x} \in \mathbf{X}$: $\left|\sum_{p \in P} h_p(\mathbf{x}) - \sum_{p \in S} w_p h_p(\mathbf{x})\right| \leq \epsilon \max_{p \in P} \frac{h_p(\mathbf{x})}{m_p} \sum_{q \in P} m_q$.*

Intuitively, the sample are weighted such that the expected sum of weights of the sample equals the total sum of weights. (The proof of the lemma is deferred to the Section 4.4.3.) To apply the lemma, we note that the bound provided depends on two terms: $\max_{p \in P} \frac{h_p(\mathbf{x})}{m_p}$ and $\sum_{q \in P} m_q$. Ideally, we want $\max_{p \in P} \frac{h_p(\mathbf{x})}{m_p}$ to be bounded and $\sum_{q \in P} m_q$ to be small. If we could choose $m_p = h_p(\mathbf{x})$, then the bound would be $\epsilon \sum_p h_p(\mathbf{x})$, which means the weighted cost of the sample approximates the original cost up to ϵ factor. However, m_p should be independent of $\mathbf{x}$. A better strategy is to choose m_p to be the upper bound for $h_p(\mathbf{x})$, then the bound is $\epsilon \sum_{q \in P} m_q$. The key now becomes to design $h_p(\mathbf{x})$ with suitable upper bounds for our problems, i.e. we should choose $h_p(\mathbf{x})$ such that its value does not vary much with $\mathbf{x}$.

We now consider applying the lemma to the k-median problem by choosing suitable $\{h_p\}$ and $\{m_p\}$. First, note that we cannot apply this lemma directly to the cost, since a suitable upper bound is not available by choosing $h_p(\mathbf{x}) := \text{cost}(p, \mathbf{x})$. We therefore consider the local approximation solutions. Let b_p denote the closest center to p, then $\text{cost}(b_p, \mathbf{x})$ will be close to $\text{cost}(p, \mathbf{x})$. We now aim to approximate the error $\sum_p [\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})]$, rather than to approximate $\sum_p \text{cost}(p, \mathbf{x})$ directly. More precisely, let $h_p(\mathbf{x}) := \text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x}) + \text{cost}(p, b_p)$, where $\text{cost}(p, b_p)$ is added so that $h_p \geq 0$. We apply the lemma to $h_p(\mathbf{x})$ and $m_p = \text{cost}(p, b_p)$. Since $0 \leq h_p \leq 2\text{cost}(p, b_p)$, the lemma bounds the difference between the cost of h_p and its sampled cost by $2\epsilon \sum_{p \in P} \text{cost}(p, b_p)$, so we have an $O(\epsilon)$-approximation.

Second, note that the difference between the cost of $\{h_p\}$ and the cost of the sampled $\{h_p\}$ is not the same as the difference between the true cost and the cost of the sampled points. We show that the gap depends only on the costs of the approximation points $\{b_p\}$, i.e.

$$\sum_{p \in P} h_p(x) - \sum_{p \in S} w_p h_p(x) \quad = \quad \sum_{p \in P} \text{cost}(p, x) - \sum_{p \in S} w_p \text{cost}(p, x) - f(\{\text{cost}(b_p, x)\}).$$

The centers are then weighted and added to the coreset, leading to an approximation good for every set of centers.

In the following, we provide a formal verification of our discussion above. We have the following lemma for k-median with $H = \{h_p : h_p(\mathbf{x}) = d(p, \mathbf{x}) - d(b_p, \mathbf{x}) + d(p, b_p), p \in P\}$.

Lemma 10. *For k-median, the output of Algorithm 8 is an ϵ-coreset with probability at least $1 - \delta$, if $t \geq \frac{c}{\epsilon^2}\left(\dim(H, (\mathcal{B}^d)^k) + \log \frac{1}{\delta}\right)$ for a sufficiently large constant c.*

Proof. We want to show that for any set of centers $\mathbf{x}$ the true cost for using these centers is well approximated by the cost on the weighted coreset. Note that our coreset has two types of points: sampled points $p \in S = \cup_{i=1}^n S_i$ with weight $w_p := \frac{\sum_{q \in P} m_q}{m_p |S|}$ and local solution centers $b \in B = \cup_{i=1}^n B_i$ with weight $w_b := |P_b| - \sum_{p \in S \cap P_b} w_p$. We use b_p to represent the nearest center to p in the local approximation solution. We use P_b to represent the set of points which have b as their closest center in the local approximation solution.

As mentioned above, we construct h_p to be the difference between the cost of p and the cost of b_p so that Lemma 9 can be applied to h_p. Note that $0 \leq h_p(x) \leq 2d(p, b_p)$ by triangle inequality, and S is sufficiently large and chosen according to weights $m_p = d(p, b_p)$, so the conditions of Lemma 9 are met. Then we have

$$\begin{aligned} D =& \left|\sum_{p\in P} h_p(\mathbf{x}) - \sum_{p\in S} w_p h_p(\mathbf{x})\right| \le \epsilon \max_{p\in P} \frac{h_p(\mathbf{x})}{m_p} \sum_{q\in P} m_q \\ \le& 2\epsilon \sum_{q\in P} m_q = 2\epsilon \sum_{p\in P} d(p, b_p) = 2\epsilon \sum_{i=1}^{n} d(P_i, B_i) \le O(\epsilon) \sum_{p\in P} d(p, \mathbf{x}) \end{aligned}$$

where the last inequality follows from the fact that B_i is a constant approximation solution for P_i.

Next, we show that the coreset is constructed such that D is exactly the difference between the true cost and the weighted cost of the coreset, which then leads to the lemma.

Note that the centers are weighted such that

$$\sum_{b\in B} w_b d(b, \mathbf{x}) = \sum_{b\in B} |P_b| d(b, \mathbf{x}) - \sum_{b\in B} \sum_{p\in S\cap P_b} w_p d(b, \mathbf{x}) = \sum_{p\in P} d(b_p, \mathbf{x}) - \sum_{p\in S} w_p d(b_p, \mathbf{x}). \quad (5)$$

Also note that $\sum_{p\in P} m_p = \sum_{p\in S} w_p m_p$, so

$$\begin{aligned} D =& \left|\sum_{p\in P} [d(p, \mathbf{x}) - d(b_p, \mathbf{x}) + m_p] - \sum_{p\in S} w_p [d(p, \mathbf{x}) - d(b_p, \mathbf{x}) + m_p]\right| \\ =& \left|\sum_{p\in P} d(p, \mathbf{x}) - \sum_{p\in S} w_p d(p, \mathbf{x}) - \left[\sum_{p\in P} d(b_p, \mathbf{x}) - \sum_{p\in S} w_p d(b_p, \mathbf{x})\right]\right|. \end{aligned} \quad (6)$$

By plugging (5) into (6), we have

$$D = \left|\sum_{p\in P} d(p, \mathbf{x}) - \sum_{p\in S} w_p d(p, \mathbf{x}) - \sum_{b\in B} w_b d(b, \mathbf{x})\right| = \left|\sum_{p\in P} d(p, \mathbf{x}) - \sum_{p\in S\cup B} w_p d(p, \mathbf{x})\right|$$

which implies the lemma. □

In [39] it is shown that $\dim(H, \left(\mathcal{B}^d\right)^k) = O(kd)$. Therefore, by Lemma 10, when $|S| \ge O\left(\frac{1}{\epsilon^2}(kd + \log\frac{1}{\delta})\right)$, the weighted cost of $S\cup B$ approximates the k-median cost of P for any set of centers, then $(S\cup B, w)$ becomes an ϵ-coreset for P. The total communication cost is bounded by $O(mn)$, since even in the most general case that every node only knows its neighbors, we can broadcast the local costs with $O(mn)$ communication (see Algorithm 10).

4.4.2 Proof of Theorem 19: k-means

We have for k-means a similar lemma that when $t = O\left(\frac{1}{\epsilon^4}(kd + \log\frac{1}{\delta}) + nk\log\frac{nk}{\delta}\right)$, the algorithm constructs an ϵ-coreset with probability at least $1-\delta$. The key idea is the same as that for k-median: we use centers $\{b_p\}$ from the local approximation solutions as an approximation to the original data $\{p\}$, and show that the error between the total cost and the weighted sample cost is approximately the error between the cost of h_p and its sampled cost (compensated by the weighted centers), which is shown to be small by Lemma 9.

The key difference between k-means and k-median is that triangle inequality applies directly to the k-median cost. In particular, for the k-median problem note that $\mathrm{cost}(b_p, p) = d(b_p, p)$ is an upper bound for the error of b_p on any set of centers, i.e. $\forall \mathbf{x} \in (\mathcal{B}^d)^k$, $d(b_p, p) \geq |d(p, \mathbf{x}) - d(b_p, \mathbf{x})| = |\mathrm{cost}(p, \mathbf{x}) - \mathrm{cost}(b_p, \mathbf{x})|$ by triangle inequality. Then we can construct $h_p(\mathbf{x}) := \mathrm{cost}(p, \mathbf{x}) - \mathrm{cost}(b_p, \mathbf{x}) + d(b_p, p)$ such that $h_p(\mathbf{x})$ is bounded. In contrast, for k-means, the error $|\mathrm{cost}(p, \mathbf{x}) - \mathrm{cost}(b_p, \mathbf{x})| = |d(p, \mathbf{x})^2 - d(b_p, \mathbf{x})^2|$ does not have such an upper bound. The main change to the analysis is that we divide the points into two categories: good points whose costs approximately satisfy the triangle inequality (up to a factor of $1/\epsilon$) and bad points. The good points for a fixed set of centers $\mathbf{x}$ are defined as

$$G(\mathbf{x}) = \{p \in P : |\mathrm{cost}(p, \mathbf{x}) - \mathrm{cost}(b_p, \mathbf{x})| \leq \Delta_p\}$$

where the upper bound is $\Delta_p = \frac{\mathrm{cost}(p, b_p)}{\epsilon}$. Good points we can bound as before. For bad points we can show that while the difference in cost may be larger than $\mathrm{cost}(p, b_p)/\epsilon$, it must still be small, namely $O(\epsilon \min\{\mathrm{cost}(p, \mathbf{x}), \mathrm{cost}(b_p, \mathbf{x})\})$.

Formally, the functions h_p are restricted to be defined only over good points:

$$h_p = \begin{cases} \mathrm{cost}(p, \mathbf{x}) - \mathrm{cost}(b_p, \mathbf{x}) + \Delta_p & \text{if } p \in G(\mathbf{x}), \\ 0 & \text{otherwise.} \end{cases}$$

Then $\sum_{p \in P} \mathrm{cost}(p, \mathbf{x}) - \sum_{p \in S \cup B} w_p \mathrm{cost}(p, \mathbf{x})$ is decomposed into three terms:

$$\sum_{p \in P} h_p(\mathbf{x}) - \sum_{p \in S} w_p h_p(\mathbf{x}) \tag{7}$$

$$+ \sum_{p \in P \setminus G(\mathbf{x})} [\mathrm{cost}(p, \mathbf{x}) - \mathrm{cost}(b_p, \mathbf{x}) + \Delta_p] \tag{8}$$

$$- \sum_{p \in S \setminus G(\mathbf{x})} w_p [\mathrm{cost}(p, \mathbf{x}) - \mathrm{cost}(b_p, \mathbf{x}) + \Delta_p] \tag{9}$$

Lemma 9 bounds (7) by $O(\epsilon)\mathrm{cost}(P, \mathbf{x})$, but we need an accuracy of ϵ^2 to compensate for the $1/\epsilon$ factor in the upper bound, resulting in a $O(1/\epsilon^4)$ factor in the sample complexity.

We begin by bounding (8). Note that for each term in (8), $|\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})| > \Delta_p$ since $p \notin G(\mathbf{x})$. Furthermore, $p \notin G(\mathbf{x})$ only when p and b_p are close to each other and far away from $\mathbf{x}$. In Lemma 11 we use this to show that $|\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})| \leq O(\epsilon) \min\{\text{cost}(p, \mathbf{x}), \text{cost}(b_p, \mathbf{x})\}$.

Using this, (8) can be bounded by $O(\epsilon) \sum_{p \in P \setminus G(\mathbf{x})} \text{cost}(p, \mathbf{x}) \leq O(\epsilon)\text{cost}(P, \mathbf{x})$.

Similarly, by the definition of Δ_p and Lemma 11, (9) is bounded by

$$\begin{aligned}
(9) \quad &\leq \sum_{p \in S \setminus G(\mathbf{x})} 2w_p|\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})| \leq O(\epsilon) \sum_{p \in S \setminus G(\mathbf{x})} w_p \, \text{cost}(b_p, \mathbf{x}) \\
&\leq O(\epsilon) \sum_{b \in B} \left(\sum_{p \in P_b \cap S} w_p \right) \text{cost}(b, \mathbf{x}).
\end{aligned}$$

Note that the expectation of $\sum_{p \in P_b \cap S} w_p$ is $|P_b|$. By a sampling argument (Lemma 12), if $t \geq O(nk \log \frac{nk}{\delta})$, then $\sum_{p \in P_b \cap S} w_p \leq 2|P_b|$. Then (9) is bounded by $O(\epsilon) \sum_{b \in B} \text{cost}(b, \mathbf{x})|P_b| = O(\epsilon) \sum_{p \in P} \text{cost}(b_p, \mathbf{x})$ where $\sum_{p \in P} \text{cost}(b_p, \mathbf{x})$ is at most a constant factor more than the optimum cost.

Since each of (7), (8), and (9) is $O(\epsilon)\text{cost}(P, \mathbf{x})$, we know that their sum is the same magnitude. Combining the above bounds, we have $\left|\text{cost}(P, \mathbf{x}) - \sum_{p \in S \cup B} w_p \text{cost}(p, \mathbf{x})\right| \leq O(\epsilon)\text{cost}(P, \mathbf{x})$. The proof is then completed by choosing a suitable ϵ, and bounding $\dim(H, (\mathcal{B}^d)^k) = O(kd)$ as in [39].

4.4.3 Proof of Lemmas

The proof of Lemma 9 follows from the analysis in [39], although not explicitly stated there. We begin with the following theorem for uniform sampling on a function space.

Theorem 20 (Theorem 6.9 in [39]). *Let F be a set of functions from $\mathbf{X}$ to $\mathcal{B}_{\geq 0}$, and let $\epsilon \in (0, 1)$. Let S be a sample of*

$$|S| = \frac{c}{\epsilon^2}(\dim(F, \mathbf{X}) + \log \frac{1}{\delta})$$

i.i.d items from F, where c is a sufficiently large constant. Then, with probability at least $1 - \delta$, for any $\mathbf{x} \in \mathbf{X}$ and any $r \geq 0$,

$$\left| \frac{\sum_{f \in F, f(\mathbf{x}) \leq r} f(\mathbf{x})}{|F|} - \frac{\sum_{f \in S, f(\mathbf{x}) \leq r} f(\mathbf{x})}{|S|} \right| \leq \epsilon r.$$

Proof of Lemma 9. Without loss of generality, assume $m_p \in \mathbf{N}^+$. Define G as follows: for each $h_p \in H$, include m_p copies of h_p/m_p in G. Then S is equivalent to a sample draw i.i.d. and uniformly at random from G. We now apply Theorem 20 on $F = G$ and $r = \max_{g\in G} g(\mathbf{x})$. By Theorem 20, we know that for any $\mathbf{x}$ and any $r \geq 0$,

$$\left|\frac{\sum_{g\in G} g(\mathbf{x})}{|G|} - \frac{\sum_{g\in S} g(\mathbf{x})}{|S|}\right| \leq \epsilon \max_{g\in G} g(\mathbf{x}). \tag{10}$$

The lemma then follows from multiplying both sides of (10) by $|G| = \sum_{p\in P} m_p$. Also note that the dimension of G is the same as that of H. □

Lemma 11. *If* $d(p,b_p)^2/\epsilon \leq |d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2|$*, then*

$$|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq 8\epsilon \min\{d(p,\mathbf{x})^2, d(b_p,\mathbf{x})^2\}.$$

Proof. We first have by triangle inequality

$$|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq d(p,b_p)[d(p,\mathbf{x}) + d(b_p,\mathbf{x})].$$

Then by $d(p,b_p)^2/\epsilon \leq |d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2|$,

$$d(p,b_p) \leq \epsilon[d(p,\mathbf{x}) + d(b_p,\mathbf{x})].$$

Therefore, we have

$$\begin{aligned}
|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| &\leq d(p,b_p)[d(p,\mathbf{x}) + d(b_p,\mathbf{x})] \leq \epsilon[d(p,\mathbf{x}) + d(b_p,\mathbf{x})]^2 \\
&\leq 2\epsilon[d(p,\mathbf{x})^2 + d(b_p,\mathbf{x})^2] \leq 2\epsilon[d(p,\mathbf{x})^2 + (d(p,\mathbf{x}) + d(p,b_p))^2] \\
&\leq 2\epsilon[d(p,\mathbf{x})^2 + 2d(p,\mathbf{x})^2 + 2d(p,b_p)^2] \leq 6\epsilon d(p,\mathbf{x})^2 + 4\epsilon d(p,b_p)^2 \\
&\leq 6\epsilon d(p,\mathbf{x})^2 + 4\epsilon^2|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2|
\end{aligned}$$

for sufficiently small ϵ. Then

$$|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq \frac{6\epsilon}{1-4\epsilon^2} d(p,\mathbf{x})^2 \leq 8\epsilon d(p,\mathbf{x})^2.$$

Similarly, we have $|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq 8\epsilon d(b_p,\mathbf{x})^2$. The lemma follows from the last two inequalities. □

Lemma 12 (Corollary 15.3 in [39]). *Let $0 < \delta < 1/2$, and $t \geq c|B| \log \frac{|B|}{\delta}$ for a sufficiently large c. Then with probability at least $1 - \delta$,*

$$\forall b \in B, \sum_{p \in P_b \cap S} w_b \leq 2|P_b|.$$

4.5 Effect of Network Topology on Communication Cost

We now apply our distributed coreset construction algorithm to distributed clustering tasks where the nodes are arranged in some arbitrary connected topology, and can only communicate with their neighbors. We propose a message passing approach for globally sharing information, and use it for collecting information for coreset construction and sharing the local portions of the coreset.The details are presented in Algorithm 9.

Algorithm 9 Distributed clustering on a graph

```
1: Input: $\{P_i\}, \{N_i\}, \mathcal{A}_\alpha$. Here $N_i$ is the neighbors of $v_i$, and $\mathcal{A}_\alpha$ is a $\alpha$-approximation
   algorithm for weighted clustering instances.
2: Round 1: on each node $v_i$
3: Construct its local portion $D_i$ of an $\epsilon/2$-coreset by Algorithm 8, using Message-Passing
   for communicating the local costs.
4: Round 2: on each node $v_i$
5: Call Message-Passing($D_i, N_i$).
6: $\mathbf{x} = \mathcal{A}_\alpha(\bigcup_j D_j)$.
7: Output: $\mathbf{x}$
```

Algorithm 10 Message-Passing(I_i, N_i)

```
1: Input: $I_i$ is the message, $N_i$ are the neighbors.
2: Let $R_i$ denote the information received
3: Initialize $R_i = \{i\}$. Send $I_i$ to all the neighbors.
4: While $R_i \neq [n]$:
   If receive message $I_j$,
   $R_i = R_i \cup \{j\}$ and send $I_j$ to all the neighbors.
```

4.5.1 General Graphs

We now present the main result for distributed clustering on graphs.

Theorem 21. *Given an α-approximation algorithm for weighted k-means (k-median respectively) as a subroutine, there exists an algorithm that with probability at least $1 - \delta$ outputs a $(1 + \epsilon)\alpha$-approximation solution for distributed k-means (k-median respectively)* *clustering. The total communication cost is* $O(m(\frac{1}{\epsilon^4}(kd+\log\frac{1}{\delta})+nk\log\frac{nk}{\delta}))$ *for* k*-means, and* $O(m(\frac{1}{\epsilon^2}(kd+\log\frac{1}{\delta})+nk))$ *for* k*-median.*

Proof. By Theorem 19, the output of Algorithm 8 is a coreset. Observe that in Algorithm 10, for any j, I_j propagates on the graph in a breadth-first-search style, so at the end every node receives I_j. This holds for all $1 \leq j \leq n$, so all nodes has a copy of the coreset at the end, and thus the output is a $(1+\epsilon)\alpha$-approximation solution.

Also observe that in Algorithm 10, for any node v_i and $j \in [n]$, v_i sends out I_j once, so the communication of v_i is $|N_i| \times \sum_{j=1}^{n} |I_j|$. The communication cost of Algorithm 10 is $O(m\sum_{j=1}^{n}|I_j|)$. Then the total communication cost of Algorithm 9 follows from the size of the coreset constructed. □

In contrast, an approach where each node constructs an ϵ-coreset for k-means and sends it to the other nodes incurs communication cost of $\tilde{O}(\frac{mnkd}{\epsilon^4})$. Our algorithm significantly reduces this.

4.5.2 Rooted Trees

Our algorithm can also be applied on a rooted tree, and compares favorably to other approaches involving coresets [88]. We can restrict message passing to operating along this tree, leading to the following theorem for this special case.

Theorem 22. *Given an* α*-approximation algorithm for weighted* k*-means (*k*-median respectively) as a subroutine, there exists an algorithm that with probability at least* $1-\delta$ *outputs a* $(1+\epsilon)\alpha$*-approximation solution for distributed* k*-means (*k*-median respectively) clustering on a rooted tree of height* h*. The total communication cost is* $O(h(\frac{1}{\epsilon^4}(kd+\log\frac{1}{\delta})+nk\log\frac{nk}{\delta}))$ *for* k*-means, and* $O(h(\frac{1}{\epsilon^2}(kd+\log\frac{1}{\delta})+nk))$ *for* k*-median.*

Proof. We can construct the distributed coreset using Algorithm 8. In the construction, the costs of the local approximation solutions are sent from every node to the root, and the sum is sent to every node by the root. After the construction, the local portions of the coreset are sent from every node to the root. A local portion D_i leads to a communication cost of $O(|D_i|h)$, so the total communication cost is $O(h\sum_{i=1}^{n}|D_i|)$. Once the coreset is constructed at the root, the α-approximation algorithm can be applied centrally, and the results can be sent back to all nodes. $\square$

Our approach improves the cost of $\tilde{O}(\frac{nh^2kd}{\epsilon^2})$ for k-median in [88] [3].

In a general graph, any rooted tree will have its height h at least as large as half the diameter. For sensors in a grid network, this implies $h = \Omega(\sqrt{n})$. In this case, our algorithm gains a significant improvement over existing algorithms.

[3] Their algorithm used coreset construction as a subroutine. The construction algorithm they used builds coreset of size $\tilde{O}(\frac{nkh}{\epsilon^d} \log |P|)$. For fair comparison, here we assume it uses state-of-the-art coreset construction.

CHAPTER V

SUBMODULAR OPTIMIZATION

5.1 Introduction

Many situations have the property of diminishing marginal returns. In economics, this is a common assumption for agent's utility. Coverage functions exhibit this property, as do cut functions in graphs. Functions on a ground set with this property are called submodular.

As submodular functions are so useful, a wide variety of forms of submodular optimization have been extensively studied. While unconstrained submodular minimization is polytime feasible, submodular maximization in general is hard. There are a wide variety of submodular optimization problems. These can involve problems addressing various types of constraints that might affect submodular optimization. In this chapter we focus on monotone submodular maximization subject to cardinality constraints.

We are also concerned with *distributed* optimization. In particular, we consider optimization problems where the objective is the sum of individual submodular functions. Note that the sum (or other submodular combination) of submodular functions is again submodular. This formulation is especially natural when each agent has a submodular utility function, and we view the sum of all utilities as the social welfare.

Clustering can be viewed as a submodular optimization problem. The distance from a point to a set of centers is a submodular function. A group of agents who have their own point (or set of points) each have a clustering cost that is a submodular function and the sum of these is also a submodular function. Thus the problem of the previous section fits into this framework of distributed submodular optimization. The primary difference is that clustering is a minimization problem rather than a maximization one.

Consider a group of people who jointly are working together throwing a party. They need to coordinate on a food budget, but each person has their own dietary preferences. How do they optimize their purchasing to optimize happiness and minimize communication?

Everyone has a utility function and we want to maximize the sum of these utility functions. This type of problem, submodular maximization subject to a cardinality constraint, is hard even in a centralized setting. Nevertheless, there are approximations; a simple greedy algorithm repeatedly picking the item with the highest marginal benefit achieves a $1 - 1/e$ approximation.

The question we address in this chapter is how to achieve such an approximation guarantee without incurring a large communication cost.

5.2 Related Work

There has been a long history of constrained submodular maximization, even restriction our attention to monotone submodular maximization subject to cardinality constraints. The classic work of Nemhauser et al. [71] showed that a simple greedy algorithm achieved a $(1 - 1/e)$ bound (and showed that this approximation factor was tight).

Our work primarily follows Badanidiyuru and Vondrak [4] who developed a thresholded greedy algorithm that does not depend on the rank.

5.3 Preliminaries

A real valued function on subsets of some ground set V is called a set function. Without loss of generality, we consider set functions where $f(\emptyset) = 0$. A set function $f : 2^V \to \mathcal{B}$ is said to be submodular if, for all $A, B \subseteq V$,

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B).$$

Equivalently, f is submodular if for all $A \subseteq B \subseteq V$ and $e \in V, e \notin B$, $f(A + \{e\}) - f(A) \geq f(B + \{e\}) - f(B)$. Intuitively this says that f has diminishing returns — adding an element to a set gives less value as the set gets larger.

A submodular function f is called monotone if for all $A \subseteq B \subseteq V$, $f(A) \leq f(B)$. In other words, adding an item into a set cannot decrease the value.

We shall often be interested in talking about the change in utility functions when adding some particular element to a set. For a submodular function f we define $f(x|S) :=$

$f(\{x\} \cup S) - f(S)$. This is sometimes called the *discrete derivative*[60, 86]. It thus makes sense to say that f is ℓ-Lipschitz if the discrete derivative is bounded by some constant ℓ.

We shall refer to each node's portion of the overall submodular function as the *utility* function of that node. Assume that each node's utility function is ℓ-Lipschitz.

Problem (Distributed Submodular Maximization with Cardinality Constraints)**.** *There is a ground set on m elements. Each of n nodes have a* personal *utility function $f_i(S)$ which is a monotone, submodular function which is ℓ-Lipschitz.*

We wish to find S which maximizes the social welfare $f(S) := \sum_{i \leq n} f_i(S)$, subject to the constraint that $|S| \leq k$. This problem is hard so we shall be content with a constant factor approximation. Additionally, we wish to minimize the communication cost *used to generate the answer.*

5.4 Our Algorithm

It is also possible to distribute the original greedy algorithm. At each step, a coordinator simply asks each person at what their relative gain is for each element in the base set. This has communication cost $m \cdot n \cdot k$, and achieves the same accuracy as the centralized greedy approximation.

We can modify the algorithm of Badanidiyuru and Vondrak [4] for cardinality constrained submodular maximization, to apply to the distributed setting. The main difference is that we estimate the overall value of adding a particular item by randomly sampling a subset of nodes.

Theorem 23. *With probability $1-\delta$, algorithm produces a $1-1/e-\epsilon$ approximation to Problem 1, with a total communication cost of $\frac{1}{\epsilon^3} m \left(\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log m\right) \cdot \frac{m}{\epsilon} \left(\log n + \log 1/\epsilon\right)$.*

The proof, like the algorithm, follows the outline of [4]. The main idea is that when we select an element to add to our set, we guarantee an increase of social welfare that is a sizable fraction of `OPT`. We need two lemmas. The first lemma probabilistically ensures that our sample value is accurate, in other words that $(1-\epsilon) f_S(x) \leq \hat{f}_S(x) \leq \frac{f_S(x)}{1-\epsilon}$ for each of our $\frac{1}{\epsilon^3} m \left(\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log m\right) \cdot \frac{m}{\epsilon} \left(\log n + \log 1/\epsilon\right)$ calls to the subroutine `SAMPLE`. The

Algorithm 11 Distributed Greedy via Thresholds

```
$S \leftarrow \emptyset$
$w \leftarrow \max_{x \in U}$ SAMPLE$(x|\emptyset)$
**while** $w \geq \epsilon/m$ **do**
  **for** $x \in U - S$ **do**
    **if** SAMPLE$(x) > w$ **then**
      $S \leftarrow S \cup \{x\}$
      **if** $|S| = k$ **then**
        **return** $S$
      **end if**
    **end if**
    $w \leftarrow w(1-\epsilon)$
  **end for**
**end while**
**return** $S$
```

Algorithm 12 SAMPLE$(x|S)$

```
$t = 0$
**for** $i$ from 1 to $\tilde{O}\left(\frac{1}{\epsilon^3}m\right)$ **do**
  Choose node $j \in N$ uniformly
  $t \leftarrow t + f_j(x|S)$
**end for**
**return** $t/O\left(\frac{1}{\epsilon^3}m\left(\log\frac{1}{\delta} + \log\frac{1}{\epsilon} + \log m\right)\right) \cdot N$
```

second lemma observes that whenever we add an element into our set, it must be reasonable compared to all the elements of OPT that we could have chosen instead. (This is analogous to Claim 3.1 of [4].)

We shall say a call to SAMPLE succeeds if the following holds:

Lemma 13. *With probability $1-\delta$, each of the $\frac{m}{\epsilon}(\log n + \log 1/\epsilon)$ calls to SAMPLE succeeds.*

This is a standard application of a multiplicative Chernoff bound. We will consider a single call to SAMPLE and show that it fails with probability at most δ/n_s, where n_s is the number of calls to SAMPLE. We prove this by a Chernoff bound, and then apply a union bound to show that all calls to sample succeed.

Note that a success here means that our approximation $\hat{f}(x|S_{i-1})$ is within a multiplicative factor of $(1+\epsilon)$ of the true value $f(x|S_{i-1})$.

Proof. Consider a single call to SAMPLE. Note that each of the $O\left(\frac{1}{\epsilon^3}m\left(\log\frac{1}{\delta} + \log\frac{1}{\epsilon} + \log m\right)\right)$ steps is an i.i.d. random variable with expected value $f(x|S)/n$. We wish to apply a multiplicative Chernoff bound, but because the bound depends on the value of $f(x|S)$ we must separate the cases where $f(x|S)$ is negligible.

The cases where $f(x|S) \geq \frac{\epsilon}{m}$ we use a multiplicative Chernoff Bound, and bound the expectation within a factor of $\left(1 + \frac{\epsilon}{m}\right)$. On the other hand, when $f(x|S)$ is quite small (as in $\leq \epsilon/m$), it suffices to guarantee that the sampled value does not exceed its expectation by more than a constant value.

□

Lemma 14. *Let us denote the ith element added by* $\mathbf{x}_i$. *Let* O *be an optimal solution. The gain of the algorithm on the ith element is* $f(\mathbf{x}_i|S_{i-1}) \geq \frac{(1-\epsilon)^3}{r} \sum_{a \in O-S} f(a|S)$.

Proof. Let the threshold when $\mathbf{x}_i$ is added be w. Since $\mathbf{x}_i$ was added we know that $\hat{f}\left(\mathbf{x}_i|S_{i-1}\right) \geq w$. Thus by Lemma 13, we know that

$$f\left(\mathbf{x}_i|S_{i-1}\right) \geq w(1-\epsilon). \tag{11}$$

On the other hand, let us consider elements $a \in O - S$. These elements were *not* chosen when the threshold was one level higher — thus $\hat{f}\left(a|S_{i-1}\right) \leq \frac{w}{1-\epsilon}$.[1]

Thus for each $a \in O - S$, we know that

$$f\left(a|S_{i-1}\right) \leq \hat{f}\left(a|S_{i-1}\right)\frac{1}{1-\epsilon} \leq \frac{w}{(1-\epsilon)^2} \tag{12}$$

Combining the above equations, we observe that $f\left(\mathbf{x}_i|S_{i-1}\right) \geq w(1-\epsilon) \geq f\left(a|S_{i-1}\right)(1-\epsilon)^3$. But this holds true for each $a \in O - S$, so we now simply average over all of the them. We conclude that

$$\begin{aligned} f\left(\mathbf{x}_i|S_{i-1}\right) &\geq \frac{1}{|O-S|}\sum_{x \in O-S}(1-\epsilon)^3 f\left(x|S_{i-1}\right) \\ &\geq \frac{(1-\epsilon)^3}{r}\sum_{x \in O-S} f\left(x|S_{i-1}\right) \end{aligned}$$

□

[1]There is a technicality here to be careful about — $\hat{f}\left(a|S_{i-1}\right)$ may have actually tried adding to a smaller set. This is true by submodularity; when we checked a, we may have done it against some set $S \subseteq S_{i-1}$. But note that $f(a|S) \leq f(a|S_{i-1})$ by submodularity.

We now can prove the desired theorem.

Theorem. Note by submodularity that $\sum_{a \in O-S} f\left(a|S_{i-1}\right) \geq f(O) - f(S_{i-1})$. Combining this observation with Lemma 14 we observe that $f\left(\mathbf{x}_i|S_{i-1}\right) \geq \frac{(1-\epsilon)^3}{r}(f(O) - f(S_{i-1}))$. But since $f\left(\mathbf{x}_i|S_{i-1}\right)$ is simply $f(S_i) - f(S_{i-1})$, this gives us a telescoping bound on $f(S_r)$ in terms of $f(O)$.

We note that

$$\begin{aligned} f(O) - f(S_i) &\leq \left(1 - \frac{(1-\epsilon)^3}{r}\right)(f(O) - f(S_{i-1})) \\ &\leq \left(1 - \frac{(1-\epsilon)^3}{r}\right)^i (f(O) - f(S_0)) \\ &= \left(1 - \frac{(1-\epsilon)^3}{r}\right)^i f(O). \end{aligned}$$

We conclude that

$$\begin{aligned} f(S_r) &\geq \left(1 - \left(1 - \frac{(1-\epsilon)^3}{r}\right)^r\right) f(O) \\ &\geq \left(1 - \left(\frac{1-3\epsilon}{r}\right)^r\right) f(O) \\ &\geq \left(1 - e^{-(1-3\epsilon)}\right) f(O) \\ &\geq (1 - 1/e - 3\epsilon) f(O) \end{aligned}$$

as desired. □

We have demonstrated that our distributed algorithm maintains the desired approximation guarantee. We now show that Algorithm 11 has the desired communication properties.

Lemma 15. *Algorithm 11 has a total communication cost of* $\frac{1}{\epsilon^3} m \left(\log\frac{1}{\delta} + \log\frac{1}{\epsilon} + \log m\right) \cdot \frac{m}{\epsilon}\left(\log n + \log 1/\epsilon\right)$.

Proof. We break this into two observations. First, communication only happens when `SAMPLE` is called. Observe that for each threshold level, there are m calls to `SAMPLE`, and there are only $\frac{1}{\epsilon}\left(\log m + \log 1/\epsilon\right)$ threshold levels. Thus in total there are $\frac{m}{\epsilon}\left(\log n + \log 1/\epsilon\right)$ calls to `SAMPLE`.

Second each call to `SAMPLE` has a communication cost of $O\left(\frac{1}{\epsilon^3}m\left(\log\frac{1}{\delta}+\log\frac{1}{\epsilon}+\log m\right)\right)$ queries to build the sample. This leads to a total communication cost of $\frac{1}{\epsilon^3}m\left(\log\frac{1}{\delta}+\log\frac{1}{\epsilon}+\log m\right)\cdot \frac{m}{\epsilon}\left(\log n+\log 1/\epsilon\right) = \frac{1}{\epsilon^3}m\left(\log\frac{1}{\delta}+\log\frac{1}{\epsilon}+\log m\right)\cdot\frac{m}{\epsilon}\left(\log n+\log 1/\epsilon\right)$. $\square$

5.5 Discussion and Future Directions

This algorithm does not depend on either the rank constraint, or the number of distributed agents. It is also possible to do a naïve distribution of greedy, that simply asks each person at each step what their relative gain is for each element in the base set. This has communication cost $m\cdot n\cdot k$, and achieves approximately the same accuracy. However, if n is large, the communication cost can be much worse.

CHAPTER VI

CONCLUSION

This thesis covered a wide variety of distributed problems.

Price of Uncertainty For Set Covering Games we improved the upper and lower bounds. The upper bound was improved to $PoU_{IR}(\epsilon, \textit{set-covering}) = (1+\epsilon)^{O(m^2)} O(\log m)$ for $\epsilon = O(\frac{1}{m})$. (Theorem 4) This does not depend on the number of players so it can be an exponential improvement over the best previous bound. (Future work can improve the case when $m = \Theta(n)$, i.e. neither the number of players nor the number of resources dominate.) We also improved the lower bound past the trivial lower bound to a polylogarithmic bound with respect to the number of players and resources. Formally, $PoU_{IR}(\epsilon, \textit{set-covering}) = \Omega(\log^p \min(m, n))$, for $\epsilon = \Theta(\frac{1}{\min(m,n)})$ and constant $p > 0$. (Theorem 7)

For Consensus Games we improved the lower bound. We show $PoU(\epsilon, consensus) = \Omega(n^2\epsilon^3)$ for $\epsilon = \Omega(\sqrt{\frac{1}{n}})$. (Theorem 2) Together with the upper bound of $PoU(\epsilon, consensus) = O(n^2\epsilon)$ (Theorem 1) we have a good understanding of the PoU in consensus games.

We also analyzed stochastic perturbations in $\lambda - \mu$ smooth games with best response dynamics. We bound the expected increase in social cost at any time step in the future. Future work can extend the analysis of stochastic perturbations to broader classes of games.

Price of Byzantine Behavior We provided a formal definition for this quantity. For consensus games we provided a tight bound on the PoB. We showed $PoB(B, consensus) = \Theta(n\sqrt{n \cdot B})$. (Theorem 3) This provides us an understanding of the limits of information spread through networks. (e.g. viral marketing) Additionally, this measure is very useful in real world settings that have to deal with bad actors.

Learning Two-Sided Disjunctions We provide algorithms that produce two-sided disjunctions which have low error rates for both labelled and unlabelled error, i.e. they are both consistent with labelled examples and compatible with the unlabelled distribution. We do this by identifying and removing non-indicator variables. For the results below, we primarily depend on k, the number of non-indicators, and a given compatibility notion $\log |C_{D,\mathcal{X}}(\epsilon)|$ which is an information theoretic lower bound on the number of labelled examples required for generalization.

Semisupervised Learning We produce a good hypothesis via a mistake-bound algorithm. As long as we can guarantee that each non-indicator has a large enough probability to be detected (which depends on ϵ_0), any mistake we detect identifies a non-indicator. Thus, we can bound the number of labelled examples as $m_l \geq \frac{1}{\epsilon_0} \log \frac{k}{\delta} + \frac{k+\log |C_{D,\mathcal{X}}(\epsilon)|}{\epsilon} \left[\log \frac{k+\log |C_{D,\mathcal{X}}(\epsilon)|}{\delta}\right]$. (Theorem 13)

Active Learning We can identify a non-indicator with only a logarithmic number of oracle queries. Formally, the number of labelling queries required is $m_q = O\left(\log |C_{D,\mathcal{X}}(\epsilon)| + k\left[\log n + \frac{1}{\epsilon}\log \frac{k}{\delta}\right]\right)$. (Theorem 14)

Both of these algorithms generalize to the case of random classification noise.

We note that the two-sided disjunctions problem can be viewed as a linear separator with a constant $\ell_\infty \ell_1$ margin. Likewise, several other open semi-supervised problems have compatibility notions that generalize to linear separators with such a margin. While other margin notions can be used to provide strong guarantees on the labelled complexity (e.g. winnow [63], perceptron [79]) we cannot bound the label complexity in terms of this margin alone. Despite this, it seems probable that there are ways to explicitly leverage this notion of margin. Future work on linear separators with $\ell_\infty \ell_1$ margins may provide an integrated view of several semi-supervised learning algorithms.

Distributed Clustering We provided an algorithm allowing agents to compute a coreset on their joint dataset with minimal communication costs. This works because the agents can jointly calculate the total weight of a global approximation by each sharing the cost of a local approximation. Formally we show that with total communication

$O(mn)$ (i.e. proportional to the number of agents n and a factor related to the topology they communicate over) the agents are able to coordinate on a coreset with size comparable to the best coreset construction in the centralized case. For k-means this is order $O(\frac{1}{\epsilon^4}(kd + \log \frac{1}{\delta}) + nk \log \frac{nk}{\delta})$ and is order $O(\frac{1}{\epsilon^2}(kd + \log \frac{1}{\delta}) + nk)$ for k-medians. (Theorem 19)

This work has been recently extended via PCA for dimensionality reduction [15] and parts have been shown to have communication bounds that are tight to within logarithmic factors[89].

Submodular Optimization We provide an algorithm which allows distributed agents to jointly maximize a shared submodular function where they each hold partial information about the global utility function, with minimal communication. Formally we allow n agents to approximately maximize the sum (or submodular combination) of their personal submodular functions subject to cardinality constraints. The communication depends only logarithmically on the number of agents.

In this thesis we have worked to understand the amount of communication needed to solve several natural distributed problems. When studying the Price of Uncertainty and Price of Byzantine Behavior we helped understand how distributed agents can handle noise in the network. Our work on semisupervised and active learning solved algorithmic problems in both the centralized and distributed settings. For the problems of clustering and submodular maximization we took a centralized algorithm and identified a key component that could be computed in a distributed manner. Specifically, the computation of a coreset only requires knowing the cost of a good approximate clustering, which can be efficiently shared. For distributed monotone submodular maximization we replaced centralized oracle queries with an approximation via sampling.

Bibliography

[1] Ackermann, H., Röglin, H., and Vöcking, B., "On the impact of combinatorial structure on congestion games," *Journal of the ACM*, vol. 55, no. 6, pp. 1–22, 2008.

[2] Anshelevich, E., Dasgupta, A., Kleinberg, J. M., Tardos, É., Wexler, T., and Roughgarden, T., "The price of stability for network design with fair cost allocation," in *FOCS*, pp. 295–304, 2004.

[3] Awerbuch, B., Azar, Y., Epstein, A., Mirrokni, V. S., and Skopalik, A., "Fast convergence to nearly optimal solutions in potential games," in *EC*, 2008.

[4] Badanidiyuru, A. and Vondrák, J., "Fast algorithms for maximizing submodular functions," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1497–1514, SIAM, 2014.

[5] Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S., "Scalable k-means++," *Proceedings of the VLDB Endowment*, vol. 5, no. 7, 2012.

[6] Balcan, M.-F., Beygelzimer, A., and Langford, J., "Agnostic active learning," in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006.

[7] Balcan, M.-F. and Blum, A., "A PAC-style model for learning from labeled and unlabeled data," in *Proceedings of the 18th Annual Conference on Computational Learning Theory (COLT)*, 2005.

[8] Balcan, M.-F., Berlind, C., Ehrlich, S., and Liang, Y., "Efficient semi-supervised and active learning of disjunctions," in *Proceedings of The 30th International Conference on Machine Learning*, pp. 633–641, 2013.

[9] Balcan, M.-F. and Blum, A., "A discriminative model for semi-supervised learning," *Journal of the ACM*, vol. 57, no. 3, 2010.

[10] Balcan, M.-F., Blum, A., Fine, S., and Mansour, Y., "Distributed learning, communication complexity, and privacy," in *Proceedings of the 25th Annual Conference on Learning Theory*, 2012.

[11] Balcan, M.-F., Blum, A., and Mansour, Y., "Improved equilibria via public service advertising," in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pp. 728–737, 2009.

[12] Balcan, M.-F., Blum, A., and Mansour, Y., "The price of uncertainty," in *EC*, 2009.

[13] Balcan, M.-F., Constantin, F., and Ehrlich, S., "The snowball effect of uncertainty in potential games," in *Internet and Network Economics*, pp. 1–12, Springer, 2011.

[14] Balcan, M.-F., Ehrlich, S., and Liang, Y., "Distributed k-means and k-median clustering on general topologies," in *Advances in Neural Information Processing Systems*, pp. 1995–2003, 2013.

[15] Balcan, M., Kanchanapally, V., Liang, Y., and Woodruff, D. P., "Improved distributed principal component analysis," *CoRR*, vol. abs/1408.5823, 2014.

[16] Ben-David, S., "A framework for statistical clustering with a constant time approximation algorithms for k-median clustering," *Learning Theory*, 2004.

[17] Berger, N., Feldman, M., Neiman, O., and Rosenthal, M., "Anarchy without Stability,"

[18] Berger, N., Feldman, M., Neiman, O., and Rosenthal, M., "Anarchy without stability," in *Symposium on Algorithmic Game Theory*, 2011. http://www.cs.princeton.edu/∼oneiman/job102.pdf.

[19] Beygelzimer, A., Hsu, D., Langford, J., and Zhang, T., "Agnostic active learning without constraints," in *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[20] Blum, A. and Balcan, M.-F., "Open problems in efficient semi-supervised PAC learning," in *Proceedings of the 20th Annual Conference on Computational Learning Theory (COLT)*, 2007.

[21] Blum, A., Hajiaghayi, M., Ligett, K., and Roth, A., "Regret minimization and the price of total anarchy," in *Proceedings of the ACM symposium on Theory of computing (STOC)*, pp. 373–382, 2008.

[22] Blum, A. and Mitchell, T., "Combining labeled and unlabeled data with co-training," in *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*, 1998.

[23] Blume, L., Easley, D., Kleinberg, J., Kleinberg, R., and Tardos, É., "Which networks are least susceptible to cascading failures?," in *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pp. 393–402, IEEE, 2011.

[24] Buchbinder, N., Lewin-Eytan, L., Naor, J., and Orda, A., "Non-cooperative cost sharing games via subsidies," in *SAGT*, pp. 337–349, 2008.

[25] Chapelle, O., Schlkopf, B., and Zien, A., *Semi-Supervised Learning.* MIT press, 2006.

[26] Charikar, M., Karloff, H., Mathieu, C., Naor, J. S., and Saks, M., "Online multicast with egalitarian cost sharing," SPAA '08, pp. 70–76, 2008.

[27] Chen, K., "On k-median clustering in high dimensions," in *SODA*, 2006.

[28] Chien, S. and Sinclair, A., "Convergence to approximate nash equilibria in congestion games," in *Proceedings of SODA, the annual ACM-SIAM symposium on Discrete algorithms*, pp. 169–178, 2007.

[29] Christodoulou, G., Mirrokni, V. S., and Sidiropoulos, A., "Convergence and approximation in potential games," in *STACS*, 2006.

[30] Considine, J., Li, F., Kollios, G., and Byers, J., "Approximate aggregation techniques for sensor databases," in *ICDE*, 2004.

[31] Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P., and others, "Spanner: Googles globally-distributed database," *OSDI*, 2012.

[32] Dasgupta, S., "Coarse sample complexity bounds for active learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[33] Dasgupta, S., "Active learning," *Encyclopedia of Machine Learning*, 2011.

[34] Dasgupta, S., Littman, M. L., and McAllester, D., "PAC generalization bounds for co-training," in *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[35] Dasgupta, S., Hsu, D., and Monteleoni, C., "A general agnostic active learning algorithm," in *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[36] Datta, S., Giannella, C., Kargupta, H., and others, "K-means clustering over peer-to-peer networks," in *Proceedings of the 8th International Workshop on High Performance and Distributed Mining*, 2005.

[37] Escoffier, B., Gourves, L., and Monnot, J., "On the impact of local taxes in a set cover game," in *Structural Information and Communication Complexity (SIROCCO)*, pp. 2–13, 2010.

[38] Fabrikant, A., Papadimitriou, C., and Talwar, K., "The complexity of pure nash equilibria," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 604–612, ACM, 2004.

[39] Feldman, D. and Langberg, M., "A unified framework for approximating and clustering data," in *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, (New York, NY, USA), pp. 569–578, ACM, 2011.

[40] Feldman, D., Sugaya, A., and Rus, D., "An effective coreset compression algorithm for large scale sensor networks," in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, ACM, 2012.

[41] Forman, G. and Zhang, B., "Distributed data clustering can be efficient and exact," *ACM SIGKDD explorations newsletter*, vol. 2, no. 2, 2000.

[42] Goemans, M., Mirrokni, V., and Vetta, A., "Sink Equilibria and Convergence," *FOCS*, 2005.

[43] Greenhill, S. and Venkatesh, S., "Distributed query processing for mobile surveillance," in *Proceedings of the 15th international conference on Multimedia*, ACM, 2007.

[44] Greenwald, M. and Khanna, S., "Power-conserving computation of order-statistics over sensor networks," in *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2004.

[45] Hanneke, S., "A bound on the label complexity of agnostic active learning," in *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007.

[46] Hanneke, S., "Teaching dimension and the complexity of active learning.," in *Proceedings of the 20th Annual Conference on Computational Learning Theory (COLT)*, 2007.

[47] Har-Peled, S. and Kushal, A., "Smaller coresets for k-median and k-means clustering," *Discrete & Computational Geometry*, vol. 37, no. 1, 2007.

[48] Har-Peled, S. and Mazumdar, S., "On coresets for k-means and k-median clustering," in *STOC*, 2004.

[49] Januzaj, E., Kriegel, H., and Pfeifle, M., "Towards effective and efficient distributed clustering," in *Workshop on Clustering Large Data Sets (ICDM)*, 2003.

[50] Joachims, T., "Transductive inference for text classification using support vector machines," in *Proceedings of the 16th International Conference on Machine Learning (ICML)*, 1999.

[51] Joachims, T., "Transductive inference for text classification using support vector machines," in *Proceedings of the 16th International Conference on Machine Learning (ICML)*, 1999.

[52] Kaariainen, M., "Generalization error bounds using unlabeled data," in *Proceedings of the 18th Annual Conference on Computational Learning Theory (COLT)*, 2005.

[53] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y., "A local search approximation algorithm for k-means clustering," in *Proceedings of the eighteenth annual symposium on Computational geometry*, 2002.

[54] Kargupta, H., Huang, W., Sivakumar, K., and Johnson, E., "Distributed clustering using collective principal component analysis," *Knowledge and Information Systems*, vol. 3, 2001.

[55] Kearns, M., "Efficient noise-tolerant learning from statistical queries," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 983–1006, 1998.

[56] Kempe, D., Kleinberg, J., and Tardos, É., "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146, ACM, 2003.

[57] Kempe, D., Kleinberg, J., and Tardos, É., "Influential nodes in a diffusion model for social networks," *Automata, Languages and Programming*, pp. 99–99, 2005.

[58] Koltchinskii, V., "Rademacher complexities and bounding the excess risk in active learning," *Journal of Machine Learning*, vol. 11, pp. 2457–2485, 2010.

[59] Koutsoupias, E. and Papadimitriou, C. H., "Worst-case equilibria," in *STACS*, pp. 404–413, 1999.

[60] Krause, A. and Golovin, D., "Submodular function maximization," *Tractability: Practical Approaches to Hard Problems*, vol. 3, p. 19, 2012.

[61] Langberg, M. and Schulman, L., "Universal ε-approximators for integrals," in *SODA*, Society for Industrial and Applied Mathematics, 2010.

[62] Li, S. and Svensson, O., "Approximating k-median via pseudo-approximation," in *STOC*, 2013.

[63] Littlestone, N., "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Machine Learning*, vol. 2, no. 4, pp. 285–318, 1988.

[64] Meir, R., Tennenholtz, M., Bachrach, Y., and Key, P., "Congestion games with agent failures," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[65] Mitra, S., Agrawal, M., Yadav, A., Carlsson, N., Eager, D., and Mahanti, A., "Characterizing web-based video sharing workloads," *ACM Transactions on the Web (TWEB)*, vol. 5, 2011.

[66] Monderer, D. and Shapley, L., "Potential games," *Games and Economic Behavior*, vol. 14, pp. 124–143, 1996.

[67] Monderer, D., Ashlagi, I., and Tennenholtz, M., "Resource selection games with unknown number of players," in *Proceedings of AAMAS'06*, pp. 819–825.

[68] Morris, S., "Contagion," *The Review of Economic Studies*, vol. 67, no. 1, pp. 57–78, 2000.

[69] Moscibroda, T., Schmid, S., and Wattenhofer, R., "When selfish meets evil: byzantine players in a virus inoculation game," in *PODC*, pp. 35–44, 2006.

[70] Moscibroda, T., Schmid, S., and Wattenhofer, R., "The price of malice: A game-theoretic framework for malicious behavior in distributed systems," *Internet Mathematics*, vol. 6, no. 2, pp. 125–155, 2009.

[71] Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L., "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[72] Neyman, A., "Correlated equilibrium and potential games," *International Journal of Game Theory*, vol. 26, no. 2, pp. 223–227, 1997.

[73] Nisan, N., Roughgarden, T., Vazirani, V., and Tados, É., *Algorithmic game theory.* Cambridge Univ Pr, 2007.

[74] Olston, C., Jiang, J., and Widom, J., "Adaptive filters for continuous queries over distributed data streams," in *ACM SIGMOD 2003*, 2003.

[75] Panagopoulou, P. and Spirakis, P., "Algorithms for pure nash equilibria in weighted congestion games," *J. Exp. Algorithmics*, vol. 11, pp. 2–7, 2006.

[76] Piliouras, G., Valla, T., and Végh, L. A., "Lp-based covering games with low price of anarchy," in *Internet and Network Economics*, pp. 184–197, Springer, 2012.

[77] Rigollet, P., "Generalized error bounds in semi-supervised classification under the cluster assumption," *Journal of Machine Learning Research*, vol. 8, 2007.

[78] Rosenberg, D. S. and Bartlett, P. L., "The rademacher complexity of co-regularized kernel classes," in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

[79] Rosenblatt, F., "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–407, 1958.

[80] Rosenthal, R., "A class of games possessing pure-strategy Nash equilibria," *Intl. Journal of Game Theory*, vol. 2, pp. 65–67, 1973.

[81] Roth, A., "The price of malice in linear congestion games," in *Proceedings of Workshop on Internet and Network Economics (WINE)*, pp. 118–125, 2008.

[82] Roughgarden, T., "Intrinsic robustness of the price of anarchy," in *Proc. of STOC*, pp. 513–522, 2009.

[83] Syrgkanis, V., "The complexity of equilibria in cost sharing games," in *WINE*, pp. 366–377, 2010.

[84] Tasoulis, D. and Vrahatis, M., "Unsupervised distributed clustering," in *IASTED International Conference on Parallel and Distributed Computing and Networks*, 2004.

[85] Valiant, L. G., "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[86] Vondrák, J., "A note on concentration of submodular functions," *arXiv preprint arXiv:1005.2791*, 2010.

[87] Voorneveld, M., "Best-response potential games," *Economics letters*, vol. 66, no. 3, pp. 289–295, 2000.

[88] Zhang, Q., Liu, J., and Wang, W., "Approximate clustering on distributed data streams," in *ICDE*, 2008.

[89] Zhang, Q., "On the communication complexity of distributed clustering," *CoRR*, vol. abs/1507.00026, 2015.

[90] Zhu, X., Ghahramani, Z., and Lafferty, J., "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003.

[91] Zhu, X., "Semi-supervised learning," *Encyclopedia of Machine Learning*, 2011.