Learning Coalition Structures with Games

Yixuan Even Xu[1]*, Chun Kai Ling[2,3], Fei Fang[3]

[1]Tsinghua University
[2]Columbia University
[3]Carnegie Mellon University
xuyx20@mails.tsinghua.edu.cn, {chunkail,feif}@cs.cmu.edu

Abstract

Coalitions naturally exist in many real-world systems involving multiple decision makers such as ridesharing, security, and online ad auctions, but the coalition structure among the agents is often unknown. We propose and study an important yet previously overseen problem – Coalition Structure Learning (CSL), where we aim to carefully design a series of games for the agents and infer the underlying coalition structure by observing their interactions in those games. We establish a lower bound on the sample complexity – defined as the number of games needed to learn the structure – of any algorithms for CSL and propose the Iterative Grouping (IG) algorithm for designing normal-form games to achieve the lower bound. We show that IG can be extended to other succinct games such as congestion games and graphical games. Moreover, we solve CSL in a more restrictive and practical setting: auctions. We show a variant of IG to solve CSL in the auction setting even if we cannot design the bidder valuations. Finally, we conduct experiments to evaluate IG in the auction setting and the results align with our theoretical analysis.

1 Introduction

Coalitions are an integral part of large, multi-agent environments. Some coalitions can lead to undesirable outcomes. For example, in ridesharing platforms (e.g., Uber, Lyft), groups of drivers sometimes deliberately and simultaneously disconnect themselves from the platform in hopes of artificially inducing a price surge which they enjoy later at the expense of the platform and riders (Hamilton 2019; Sweeney 2019; Dowling 2023), sparking studies on mechanisms to discourage such behaviors (Tripathy, Bai, and Heese 2022). In security domains, coordinated attacks are often more difficult to mitigate compared to those conducted in isolation. (Jena, Ghosh, and Koley 2021; Lakshminarayana, Belmega, and Poor 2019). On the other hand, coalitions are common and crucial to the proper functioning of real-world societies.

Ultimately, knowing the underlying coalition structure in such environments can lead to more accurate game models, more robust strategies, or the construction of better welfare-maximizing mechanisms. However, unlike payoffs, it is often not known apriori which coalitions (if any) exist. As such, we propose the *Coalition Structure Learning* (CSL) problem, where we actively put agents through a small set of carefully designed games and infer the underlying coalition structure by observing their behavior.

We stress the difference between our work and cooperative game theory. Our work identifies coalition structures by exploiting the differences in interactions between agents and is separate from the study of underlying mechanisms ensuring the stability of the said coalitions.

In this paper, we assume members in a coalition secretly share their individual utilities, i.e., they act as a joint agent whose utility equals the sum of the individual utilities of its members. Crucially, this difference in behavior allows us to detect coalitions. Consider the game shown in Fig. 1a, a variant of the classic Prisoner's Dilemma. Here, the only Nash Equilibrium (NE) is for both agents to **D**efect. However, if they are in a coalition, they behave collectively as a single agent with payoffs shown in Fig. 1b. From the coalition's perspective, it is rational for both agents to **C**ooperate as it maximizes the sum of both agent's payoff.

	C_y	D_y
C_x	$(3,3)$	$(0,5)$
D_x	$(5,0)$	$(\mathbf{1},\mathbf{1})$

(a) Not in a coalition

$\mathrm{C}_x\mathrm{C}_y$	$\mathrm{C}_x\mathrm{D}_y$	$\mathrm{D}_x\mathrm{C}_y$	$\mathrm{D}_x\mathrm{D}_y$
$\mathbf{3+3}$	$0+5$	$5+0$	$1+1$

(b) In a coalition

Figure 1: A variant of Prisoner's dilemma when agents x and y are (b) in and (a) not in a coalition. Bolded cells are the (unique) Nash Equilibria.

More generally, we have a set $N = \{1, 2, \ldots, n\}$ of n strategic agents[1], divided into m separate coalitions. A coalition $S \subseteq N$ is a nonempty subset of the agents, in which the agents coordinate with each other. A coalition structure of the agents is represented by a partition $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of N, where $S_1, S_2, \ldots, S_m$ are mutually disjoint coalitions and $\bigcup_{i=1}^m S_i = N$. Note that some of the coalitions might be singletons. We use $[i]_\mathcal{S}$ to denote the coalition that agent i belongs to under $\mathcal{S}$. If $[i]_\mathcal{S} = \{i\}$ for each $i \in N$, we recover the regular game setting.

*This work was done when Xu was a visiting intern at CMU.

[1]We provide a list of key notations in Appendix A.

In CSL, both m and $\mathcal{S}$ are unknown, and the goal is to recover them by observing how the agents interact with each other in a series of designed games. At each timestep, we present a game $\mathcal{G}$ to the agents and make an observation $\mathcal{O}$ about the equilibria in $\mathcal{G}$. As shown in Fig. 1, different coalition structures will lead to different sets of equilibria, which makes CSL possible to solve. We restrict $\mathcal{O}$ to be a single-bit oracle, indicating whether a pre-specified strategy profile Σ is a Nash Equilibrium of $\mathcal{G}$. This is the simplest observation to make and can be implemented in practice by presenting Σ as a default strategy profile to the agents and observing whether *any* agent deviates from it.

We define the *sample complexity* of an algorithm on a CSL instance as the number of games it presents to agents before the correct coalition structure is learned. We are interested in algorithms with low sample complexity.

In this paper, we thoroughly study CSL with the single-bit observation oracle $\mathcal{O}$. In many real-world settings, there will be restrictions on what kind of games can be designed and presented to the agents. Therefore, we study CSL under various settings of the class of games that $\mathcal{G}$ belongs to. Specifically, we make the following contributions: **(1).** We propose and formally model the CSL problem. **(2).** We show a lower bound of sample complexity as a function of the number of agents n for algorithms solving CSL (Theorem 3.1). **(3).** We propose our Iterative Grouping (IG) algorithm for solving CSL when $\mathcal{G}$ is restricted to normal form games (Algorithm 1) and show that it achieves the optimal sample complexity up to low order terms (Theorem 3.2). **(4).** We extend IG to solve CSL with congestion games and graphical games, again with optimal sample complexity (Section 3.4). **(5).** We propose AuctionCSL, a variant of CSL in the grounded setting of second-price auctions with personalized reserve prices, and extend IG to solve AuctionCSL (Section 4). **(6).** We extensively conduct experiments to evaluate IG in the auction setting (Section 5). The experiments align with our theoretical results, showing that IG is a practical approach to AuctionCSL. Below we summarize the theoretical results of this paper in Table 1.

Setting	Sample Complexity	Section
Lower Bound	$(1-o(1))n\log_2 n$	Section 3.1
Normal Form	$n\log_2 n + 3n$	Section 3.3
Congestion	$n\log_2 n + 3n$	Section 3.4
Graphical	$n\log_2 n + 3n$	Appendix C
Auction	$(4.16 + o(1))n\log_2 n$	Section 4

Table 1: Summary of theoretical results.

2 Related Work

In recent years, there has been significant interest in the learning of games. One such direction is *Inverse Game Theory*, which seeks to compute game parameters (e.g., agent utilities, chance) that give rise to a particular empirically observed equilibrium (Waugh, Ziebart, and Bagnell 2011; Kuleshov and Schrijvers 2015; Ling, Fang, and Kolter 2018; Geiger and Straehle 2021; Peng et al. 2019; Letchford, Conitzer, and Munagala 2009). In an "active" setting closer to our work, Balcan et al. (2015); Haghtalab et al. (2016) show that attacker utilities in Stackelberg security games may be learned by observing best-responses to chosen defender strategies. More broadly, the field of *Empirical Game-Theoretic Analysis* reasons about games and their structure by interleaving game simulation and analysis (Wellman 2006). Another related direction is given by Athey and Haile (2002), who identify different auctions based on winning bids or bidders. Recent work by Kenton et al. (2023) distinguishes between agents and the environment by extending techniques from causal inference. In all of these works, the focus is to learn agent payoffs and other game parameters (e.g., chance probabilities, item valuations, and distributions), assuming that agents and any coalitions are pre-specified. In contrast, CSL learns *coalition structures* given the freedom to design agent payoffs or other game parameters. Finally, Mazrooei, Archibald, and Bowling (2013) and Bonjour, Aggarwal, and Bhargava (2022) detect the existence of a single coalition, but not the entire coalition structure in multiplayer games.

3 CSL with Normal Form Games

In this section, we present how to solve the CSL problem when $\mathcal{G}$ is restricted to the set of all normal form games. We assume in this section that we have the power to design the whole game matrix. This section demonstrates the main idea of the paper, which will be recurring in more complicated and restricted settings in Section 4.

3.1 Lower Bound of Sample Complexity

We start our investigation with a lower bound of the sample complexity of any algorithm that solves the CSL problem. It serves as a reference for designing future algorithms.

Theorem 3.1. *An algorithm solving the CSL problem has a sample complexity of at least* $n\log_2 n - O(n\log_2\log_2 n)$.

Proof of Theorem 3.1: For every game $\mathcal{G}$ presented to the agents, we get at most 1 bit of information from $\mathcal{O}$. The number of possible partitions of N is the Bell number B_n. Therefore, to distinguish between all possible partitions, we need at least $\lceil\log_2 B_n\rceil = n\log_2 n - O(n\log_2\log_2 n)$ bits of information, which follows from the asymptotic expression of Bell number established in De Bruijn (1981). ■

3.2 Pairwise Testing via Normal-Form Gadgets

Let $\mathcal{S}^*$ be the ground truth coalition structure. It is useful to consider the problem of determining if a given pair of agents (x, y) are in the same coalition, i.e., $[x]_{\mathcal{S}^*} = [y]_{\mathcal{S}^*}$. The solution to this subproblem is given by a *normal-form gadget* game inspired by Fig. 1, and forms the building block toward our eventual Iterative Grouping algorithm.

Definition 3.1. *A game-strategy pair* $(\mathcal{G}, \Sigma)$ *is a* n*-player normal form game with* Σ *as a* ***default strategy profile*** *in* $\mathcal{G}$.

Definition 3.2. *A* ***normal form gadget*** $\mathcal{N}(x, y) = (\mathcal{G}, \Sigma)$ *is a game-strategy pair where players* $i \in N \setminus \{x, y\}$ *are dummies with one action* D_i *and recieve* 0 *utility. Players* x *and* y *have actions* $\{\mathrm{C}_x, \mathrm{D}_x\}$ *and* $\{\mathrm{C}_y, \mathrm{D}_y\}$ *and utilities shown in Fig. 1a. The* ***default strategy profile*** *is* $\Sigma = (\mathrm{D}_1, \dots, \mathrm{D}_n)$.

Lemma 3.1. *The default strategy profile of $\mathcal{N}(x, y)$ is a Nash Equilibrium if and only if $[x]_{\mathcal{S}^*} \neq [y]_{\mathcal{S}^*}$.*

Proof of Lemma 3.1: If x and y are in the same coalition, they act as a joint player with utility equal to the sum of their individual utilities. Then, by deviating to $(\mathrm{C}_x, \mathrm{C}_y)$, the utility of the joint player will increase from 2 to 6. Thus the default strategy profile is not a Nash Equilibrium. If x and y are in different coalitions, then the unilateral deviation of either the coalition of x or y will not increase their utility. Thus the default strategy profile is a Nash Equilibrium. ■

We remark that the game in Fig. 1a is not the only game that can be used to construct the normal form gadget in Definition 3.2. A game with a unique NE from which agents have incentives to deviate when they are in the same coalition would serve the purpose. Definition 3.2 and Lemma 3.1 shows how to detect pairwise coalition. With Lemma 3.1, we can already solve CSL with a sample complexity of $\frac{1}{2}n(n-1)$ by querying the observation oracle $\mathcal{O}(\mathcal{N}(x, y))$ for all $1 \leq x < y \leq n$. However, we can do better by checking multiple agent pairs at the same time, as detailed next.

3.3 The Iterative Grouping Algorithm

Our Iterative Grouping (IG) algorithm solves CSL with a sample complexity matching the bound in Theorem 3.1. IG begins with an initial coalition structure where each agent is in a separate coalition. Then, for agent i, IG iteratively tries to find another agent j within i's coalition. If it finds such an agent, it merges i and j's coalitions. Otherwise, it finalizes i's coalition and moves on to the next agent. In either case, the number of unfinalized coalitions decreases. Therefore, IG will eventually find the correct coalition structure.

To find such an agent j, IG uses a method similar to binary search. Specifically, we will introduce in Lemma 3.2 a way that allows us to use the observation of a *single* game to determine for a set $T \subseteq N$, whether there is an agent $j \in T$ that is also within i's coalition, i.e., whether $T \cap [i]_{\mathcal{S}^*} \neq \varnothing$. If so, we bisect T into two sets T_α and T_β and use another game to determine which of the two sets j is in. We then repeat this process recursively to locate j efficiently.

With that in mind, we proceed to describe IG formally. We start by defining the product of game-strategy pairs, which returns a game equivalent to *playing the two games separately with utilities of each player summed*, as well as a product of default strategy profiles for each player.

Definition 3.3. *Let $\sigma_1 = (c_1, \ldots, c_{k_1}), \sigma_2 = (d_1, \ldots, d_{k_2})$ be two mixed strategies over the sets of actions $A = \{a_1, \ldots, a_{k_1}\}, B = \{b_1, \ldots, b_{k_2}\}$ respectively, where c_θ, d_η are the probabilities of choosing a_θ, b_η respectively. The* ***product*** *of σ_1 and σ_2 is a mixed strategy $\sigma_1 \times \sigma_2$ over $A \times B$, where the probability of choosing (a_θ, b_η) is $c_\theta d_\eta$.*

Definition 3.4. *Let $(\mathcal{G}_1, \Sigma_1), (\mathcal{G}_2, \Sigma_2)$ be two game-strategy pairs where $A_{x,i}, u_{x,i}$ are the action sets and utility function of player i in $\mathcal{G}_x$ respectively for $x \in \{1, 2\}$. Let $\Sigma_1 = (\sigma_{1,i})_{i \in N}$ and $\Sigma_2 = (\sigma_{2,i})_{i \in N}$. The* ***product*** *of $(\mathcal{G}_1, \Sigma_1)$ and $(\mathcal{G}_2, \Sigma_2)$ is a game-strategy pair $(\mathcal{G}_p, \Sigma_p)$. Here, $\mathcal{G}_p$ is a normal form game with action set $A_{1,i} \times A_{2,i}$ and utility function $u_{1,i} + u_{2,i}$ for each player $i \in N$. $\Sigma_p = (\sigma_{1,i} \times \sigma_{2,i})_{i \in N}$.*

Then, by querying the observation oracle for the product of several games, we will get an aggregated observation. We formalize this idea in the following lemma.

Lemma 3.2. *Let $\{\mathcal{N}(x_\theta, y_\theta) = (\mathcal{G}_\theta, \Sigma_\theta) \mid \theta \in \{1, \ldots, k\}\}$ be a set of k normal form gadgets. The default strategy profile of $\mathcal{N}(x_1, y_1) \times \cdots \times \mathcal{N}(x_k, y_k)$ is a Nash Equilibrium if and only if for each $\theta \in \{1, 2, \ldots, k\}$, $[x_\theta]_{\mathcal{S}^*} \neq [y_\theta]_{\mathcal{S}^*}$.*

Proof of Lemma 3.2: By Definition 3.4, playing the product game is equivalent to separately playing $\mathcal{G}_1, \ldots, \mathcal{G}_k$, and sum up the resulting utilities of each player. Therefore, the default strategy profile of the product is a Nash Equilibrium if and only if the default strategy profile of each $\mathcal{G}_i$ is a Nash Equilibrium. Applying Lemma 3.1 completes the proof. ■

With Lemma 3.2, we can design a more efficient Iterative Grouping algorithm for the CSL problem (Algorithm 1).

Algorithm 1: Iterative Grouping (IG)

Input: The number of agents n and an observation oracle $\mathcal{O}$
Output: A coalition structure $\mathcal{S}$ of the agents

```
1: Let S ← {{1}, {2}, ..., {n}}.
2: for i ∈ N do
3:     while O(∏_{[j]_S ≠ [i]_S} N(i, j)) = false do
4:         Let T ← {j ∈ N | [j]_S ≠ [i]_S}.
5:         while |T| > 1 do
6:             Partition T into T_α, T_β where ||T_α| − |T_β|| ≤ 1.
7:             if O(∏_{j ∈ T_α} N(i, j)) = false then
8:                 Let T ← T_α.
9:             else
10:                Let T ← T_β.
11:        Let j ← the only element in T.
12:        Merge [i]_S and [j]_S in S.
13: return S.
```

IG (Algorithm 1) starts with the initial coalition structure $\mathcal{S} = \{\{1\}, \{2\}, \ldots, \{n\}\}$, where each agent is in a separate coalition (Line 1). In each iteration of the outer for loop (Lines 3 to 12), we consider an agent i and try to find all agents in i's coalition $[i]_{\mathcal{S}^*}$, where $\mathcal{S}^*$ is the ground truth coalition structure. In Line 3, we present a game $\prod_{[j]_{\mathcal{S}} \neq [i]_{\mathcal{S}}} \mathcal{N}(i, j)$ to the agents, where we concurrently ask each agent j that is not currently recognized as in i's coalition $[i]_{\mathcal{S}}$ to play the normal form gadget $\mathcal{N}(i, j)$ with i. If the default strategy profile in this game is not a Nash Equilibrium (Line 3), then according to Lemma 3.2, there must be an agent outside of $[i]_{\mathcal{S}}$ that is in the same coalition with i. We use binary search (Lines 4 to 10) to locate this agent j (Fig. 2) and merge i and j's coalitions (Lines 11 to 12). This is repeated until all players in i's coalition are found (Fig. 3). Repeating this for all players $i \in N$ guarantees we get $\mathcal{S} = \mathcal{S}^*$ once IG terminates.

Theorem 3.2. *IG solves the CSL problem with a sample complexity upper bounded by $n \log_2 n + 3n$.*

The proof of Theorem 3.2 is deferred to Appendix B.1. Combined with Theorem 3.1, Theorem 3.2 shows that IG solves the CSL problem with optimal sample complexity and a matching constant up to low order terms.

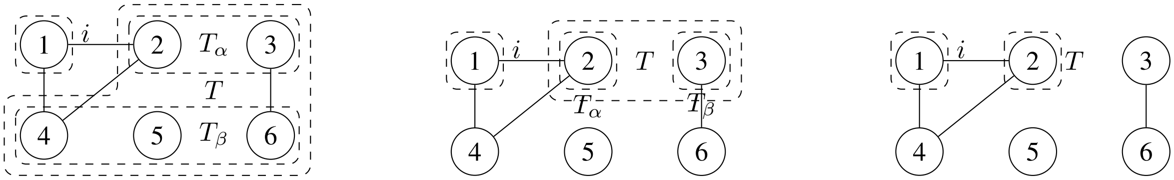


Figure 2: Example of the binary search process in Algorithm 1 (Lines 4 to 11). The ground truth coalition structure is $\mathcal{S}^* = \{\{1, 2, 4\}, \{3, 6\}, \{5\}\}$ as shown by the solid lines. The algorithm is trying to find an agent in agent 1's coalition. At first $T = \{2, 3, 4, 5, 6\}$ and is partitioned into $T_\alpha = \{2, 3\}$ and $T_\beta = \{4, 5, 6\}$ (left). As T_α contains an agent in 1's coalition, T is replaced by T_α and then partitioned into $T_\alpha = \{2\}$ and $T_\beta = \{3\}$ (middle). Then, as T_α still contains an agent in 1's coalition, T is replaced by $T_\alpha = \{2\}$ and we find an agent 2 in agent 1's coalition (right).

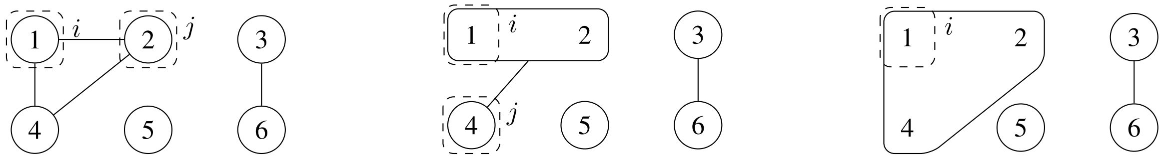
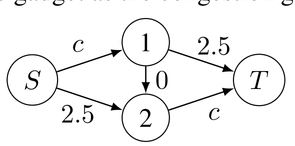


Figure 3: Example of one iteration of the outer for loop in Algorithm 1 (Lines 3 to 12). The ground truth coalition structure is $\mathcal{S}^* = \{\{1, 2, 4\}, \{3, 6\}, \{5\}\}$ as shown by the solid lines. The algorithm is trying to find all agents in agent 1's coalition. After finding agent 2 in agent 1's coalition, the algorithm merges their coalitions $\{1\}, \{2\}$ into one coalition $\{1, 2\}$ (left). Then, the algorithm finds agent 4 in agents 1 and 2's coalition and merges their coalitions $\{1, 2\}$ and $\{4\}$ into one coalition $\{1, 2, 4\}$ (middle). Finally, the algorithm confirms that agent 1's coalition is finalized (right).

3.4 Extension to Other Succinct Games

IG solves CSL with normal form games. However, sometimes there are external restrictions on what kind of games we can design and present to the agents, forbidding us from using general normal form games. Thus, in this subsection, we briefly discuss how to extend IG to other succinct games, like congestion games and graphical games.

CSL with congestion games. IG can also be extended to congestion games (Rosenthal 1973) with a modified gadget construction. For a pair of players x and y, we define the congestion game gadget as the congestion game below.



This game is a variant of the well-known Braess's paradox (Braess 1968). In this game, both players want to go from S to T. The costs of the edges are annotated on the graph where c denotes the number of players going through the edge. Let Σ denote the strategy profile where both players go through $S \to 1 \to 2 \to T$. We can see that Σ is a Nash Equilibrium if and only if x and y are in different coalitions. This is exactly what we have in Lemma 3.1. Moreover, the products of congestion games can also be represented as a congestion game. Therefore, we can use this gadget to replace $\mathcal{N}(x, y)$ in Algorithm 1 and solve the CSL problem with congestion games. The sample complexity upper bound of this algorithm is $n \log_2 n + 3n$ as well.

CSL with graphical games. A graphical game (Kearns, Littman, and Singh 2001) is represented by a graph G, where each vertex denotes a player. There is an edge between a pair of vertices x and y if and only if their utilities are dependent on each other's strategy. To limit the size of the representation of a graphical game, a common way is to limit the maximum vertex degree d in G. We show that with a slight modification, IG can be extended to solve the CSL problem with graphical games of maximum vertex degree $d = 1$ with the same sample complexity upper bound $n \log_2 n + 3n$. The details are deferred to Appendix C.

4 CSL with Auctions

We now pivot from classic games to a more practical class of games: second-price auctions with personalized reserves (Paes Leme, Pal, and Vassilvitskii 2016). Collusion of multiple agents in auctions has already been extensively observed (see, e.g., Milgrom 2004). The auction mechanisms can be exploited if these coordinated bidders deviate simultaneously. Thus, it is important to study the CSL problem with auctions. We refer to this variant of CSL as AuctionCSL.

In such an auction, each agent i has a private value v_i for the item being auctioned and a personalized reserve price r_i. Each agent i submits a bid b_i to the auction, after which the auction will choose an agent with the highest bid $i^* \in \arg\max_{i \in N}\{b_i\}$ and offer the item to i^* with price $p = \max\{r_{i^*}, \max_{i \neq i^*}\{b_i\}\}$. The agent i^* can choose to accept or reject the offer. If i^* rejects, the auction ends with no transaction. Otherwise, i^* pays p and gets the item. The item is then redistributed within i^*'s coalition to the agent with the highest private valuation for maximum coalition utility.

In this section, we consider an online auction setting where we play the role of an auctioneer. Our goal is to recover the coalition structure $\mathcal{S}^*$. As the values $\{v_i\}$ are determined by each agent's valuation for the item being auctioned, we study the setting where we can only design the reserve prices $\{r_i\}$. We assume that a stream of items will arrive to be auctioned, whose values $\{v_i\}$ are randomly generated each time, and we have no power to design them. However, we assume that we know $\{v_i\}$ before designing the reserve prices $\{r_i\}$: this happens when we are sufficiently acquainted with the agents, so we can estimate their values given a certain item. We fix the default strategy profile of the agents as truthful bidding, i.e., $b_i = v_i$ for all $i \in N$.

4.1 Group Testing via Auction Gadgets

Inspired by Algorithm 1, we still would like a way to tell whether there is an agent j inside a set T that is in the same coalition with another agent i using the result of a single auction. However, since the product of two auctions is no longer an auction (see Appendix B.6), the same method as in Section 3 is not appropriate. Therefore, we need to design a new gadget for auctions. The main idea remains the same.

Definition 4.1. *Let* $\mathbf{v} \in [0,1]^n$ *and* $T \subseteq N$. *An* ***auction gadget*** $\mathcal{A}(\mathbf{v}, T)$ *is a second price auction with personalized reserves where the values of the agents are* $\mathbf{v}$. *Let* v_{max} *and* v_{smax} *be the maximum and second maximum value among* $\mathbf{v}$ *respectively. The reserve prices of the agents are defined as*

$$r_i = \begin{cases} v_{\text{smax}} & (i \in T) \\ v_{\text{max}} & (i \notin T). \end{cases}$$

The ***default strategy profile*** *is bidding* $b_i = v_i$ *for all* $i \in N$.

We similarly establish the following connection between the result of an auction gadget and the coalition structure.

Lemma 4.1. *Let* $\mathbf{v} \in [0,1]^n$ *be a vector such that* v_i *is the unique maximum. Let* $T \subseteq N \backslash \{i\}$. *Then bidding truthfully in* $\mathcal{A}(\mathbf{v}, T)$ *is a Nash Equilibrium if and only if* $[i]_{\mathcal{S}^*} \neq [j]_{\mathcal{S}^*}$ *(i.e.,* i *and* j *are in different coalitions) for all* $j \in T$.

Proof of Lemma 4.1: Let $v_{\text{max}} = v_i$, $v_{\text{smax}} = \max_{j \neq i}\{v_j\}$. If $\exists j \in T$, such that i and j are in the same coalition, then they can jointly deviate by bidding $b_i = v_{\text{smax}}$ and $b_j = v_{\text{max}}$. In this way, j wins the auction with price $p = v_{\text{smax}}$. j can accept the item with this price, and redistribute it to i. The total utility of i and j's coalition increases from 0 to $v_{\text{max}} - v_{\text{smax}}$. Thus bidding truthfully is not a Nash Equilibrium. If $\forall j \in T$, i and j are in different coalitions, then (i) the unilateral deviation of i's coalition cannot lead to positive utility as all members in this coalition have a reserve price of v_{max} (ii) the unilateral deviation of any other coalitions cannot lead to positive utility as the maximum value among them is v_{smax}, and the reserve prices of them is at least v_{smax}. Thus bidding truthfully is a Nash Equilibrium. ■

From Lemma 4.1 and Lemma 3.2, we can see that auction gadgets are analogous to normal form gadgets. Assuming we have the freedom to design valuation vectors $\mathbf{v} \in [0,1]^n$ for an auctioned item, then $\mathcal{A}(\mathbf{v}, T)$ may be used to determine if an agent in T that is also in $[i]_{\mathcal{S}^*}$. This yields an algorithm similar to IG (Algorithm 1) for solving AuctionCSL under this simplifying assumption. We describe this algorithm and its theoretical guarantees in Appendix D.

4.2 IG under Auctions with Random Valuations

In real auctions, valuations of items are beyond our control. We model this more realistic setting by assuming that the values are drawn from an item pool $\mathcal{V}$, which is a distribution $\mathcal{U}[0,1]^n$ over $\mathbb{R}^n$. Intuitively, the randomness of the values makes CSL in this setting significantly more challenging than the normal form game setting, as we cannot guarantee progress of the algorithm if we get an unlucky draw of the values. For example, if the item has 0 value for all agents, then truthful bidding will always be a Nash Equilibrium no matter what the reserve prices are. This suggests that we can at best hope for a guarantee on the expected sample complexity. We design AuctionIG for this setting.

Algorithm 2: IG with Auctions (AuctionIG)

Input: The number of agents n and an observation oracle $\mathcal{O}$
Output: A coalition structure $\mathcal{S}$ of the agents

```
1: Let S ← {{1}, {2}, ..., {n}}.
2: Let T_i ← ∅ for all i ∈ N.
3: Let C_i ← 0 for all i ∈ N.
4: Let T_finalized ← ∅.
5: while T_finalized ≠ N do
6:     Get v ~ V.
7:     Let x ← arg max_{i∈N}{v_i} and C_x ← C_x + 1.
8:     if T_x = ∅ then
9:         if O(A(v, N \ [x]_S)) = false then
10:            Let T_i ← N \ [x]_S for all i ∈ [x]_S.
11:        else
12:            Let T_finalized ← T_finalized ∪ [x]_S.
13:    else
14:        Partition T_x into T_α, T_β where ||T_α| − |T_β|| ≤ 1.
15:        if O(A(v, T_α)) = false then
16:            Let T_i ← T_α for all i ∈ [x]_S.
17:        else
18:            Let T_i ← T_β for all i ∈ [x]_S.
19:    if |T_x| = 1 then
20:        Let y ← the only element in T_x.
21:        Merge [x]_S and [y]_S in S.
22:        Let T_i ← ∅ for all i ∈ [x]_S.
23: return S.
```

The main idea of AuctionIG is still similar to IG (Algorithm 1). For agent x, we try to iteratively find other agents in x's coalition $[x]_{\mathcal{S}^*}$ using binary search. However, as we do not have control over which agent has the largest value, we cannot do this sequentially for each agent as in IG. Instead, we run multiple instances of binary search in parallel, each progressing depending on which item is drawn.

In AuctionIG, for each $i \in N$ we maintain T_i as a set containing another agent in i's coalition (Line 2), C_i as the number of times v_i has appeared as the largest value in $\mathbf{v}$ (Line 3), and $T_{\text{finalized}}$ as the set of agents whose coalitions have been finalized (Line 4). Each time we draw an item $\mathbf{v}$ from $\mathcal{V}$, we find the agent x with the largest value (Lines 6 to 7), and try to proceed with the binary search to expand x's coalition. If $T_x = \varnothing$, then we should start a new binary search for x's coalition (Lines 9 to 12). We first check whether there is an agent in x's coalition in $N \setminus [x]_{\mathcal{S}}$. If so, we set T_i to $N \setminus [x]_{\mathcal{S}}$ for all $i \in [x]_{\mathcal{S}}$; otherwise, we know that x's coalition is finalized, and we add the entire coalition to $T_{\text{finalized}}$ (Line 12). If $T_x \neq \varnothing$, then we are in the middle of a binary search for x's coalition (Lines 13 to 18). We partition T_x into T_α and T_β and check whether there is an agent in x's coalition in T_α. If so, we set T_i to T_α for all $i \in [x]_\mathcal{S}$; otherwise, we set T_i to T_β for all $i \in [x]_\mathcal{S}$. If $|T_x| = 1$, then we have found another agent y in x's coalition (Lines 20 to 22). We merge their coalitions and set T_i to $\varnothing$ for all $i \in [x]_\mathcal{S}$, indicating that binary search should be restarted for this coalition. The outermost loop runs until $T_{\text{finalized}} = N$, which means that we have finalized the coalitions of all agents.

To analyze AuctionIG, we utilize the invariants in the following Lemma, whose proof is deferred to Appendix B.2.

Lemma 4.2. *Let $\mathcal{S}^*$ be the correct coalition structure. The following holds throughout the execution of AuctionIG.*

(a) $[i]_\mathcal{S} \subseteq [i]_{\mathcal{S}^*}, \forall i \in N$.
(b) $[i]_\mathcal{S} = [i]_{\mathcal{S}^*}, \forall i \in T_{\text{finalized}}$.
(c) $T_i = T_j$ if $[i]_\mathcal{S} = [j]_\mathcal{S}$.
(d) $\exists j \in T_i$ such that $j \in [i]_{\mathcal{S}^*} \setminus [i]_\mathcal{S}$ if $T_i \neq \varnothing$.

Next, we show a termination condition for AuctionIG.

Lemma 4.3. *AuctionIG terminates no later than the time when $C_i \geq 2\log_2 n + 4$ holds for all $i \in N$.*

We will prove Lemma 4.3 in Appendix B.3. To sketch the proof, we define $\mathcal{S}_i = \{[j]_\mathcal{S} \mid j \in [i]_{\mathcal{S}^*}\}$ and

$$f(T) = \begin{cases} \lceil \log_2 n \rceil + 1 & (T = \varnothing) \\ \lceil \log_2 |T| \rceil & (T \neq \varnothing). \end{cases}$$

According to Lemma 4.2 (c), we can unambiguously use T_S to denote T_x for any $x \in S \in \mathcal{S}_i$ and define the potential function $\Phi_i(\mathbf{T}, \mathcal{S}) = \lceil \log_2 n \rceil \cdot |\mathcal{S}_i| + \sum_{S \in \mathcal{S}_i} f(T_S)$, where $\mathbf{T}$ is the vector of all T_i. Intuitively, the potential function characterizes the remaining progress associated with agent i, where $\lceil \log_2 n \rceil \cdot |\mathcal{S}_i|$ adds $\lceil \log_2 n \rceil$ for each unmerged coalition in $\mathcal{S}_i$ and $\sum_{S \in \mathcal{S}_i} f(T_S)$ adds $\lceil \log_2 |T_S| \rceil$ for each coalition $S \in \mathcal{S}_i$, indicating the remaining steps in the binary search. To complete the proof, we show that $\Phi_i(\mathbf{T}, \mathcal{S})$ decreases by at least 1 after any C_j for $j \in [i]_{\mathcal{S}^*}$ increases.

Lemma 4.3 shows that we will have finalized the coalition structures when we have gotten for each agent i, $2\log_2 n + 4$ items that are most valuable to i. This connects the sample complexity of AuctionIG to a well-studied problem in statistics, the coupon collector's problem (Newman 1960; Erdős and Rényi 1961). In this problem, there are n types of coupons, and each time we draw a coupon, we get a coupon of a uniformly random type. We want to collect k sets of coupons, where each set contains one coupon of each type. The coupon collector's problem asks for the expected number of draws needed to collect k sets of coupons $T_{\text{ccp}}(n, k)$.

Lemma 4.3 demonstrates that the sample complexity of AuctionIG is upper bounded by $T_{\text{ccp}}(n, 2\log_2 n + 4)$. Combining this with the result of Papanicolaou and Doumas (2020) from the coupon collector's problem's literature, we have Theorem 4.1 with its proof deferred to Appendix B.4.

Theorem 4.1. *AuctionIG solves AuctionCSL with **expected** sample complexity upper bounded by $(4.16 + o(1))n \log_2 n$.*

Using Markov's inequality, we can also transform Theorem 4.1 into a PAC learning type of result as below.

Corollary 4.1 (PAC Complexity)**.** *For any $\delta \in (0, 1)$, AuctionIG correctly learns the coalition structure with probability at least $1 - \delta$ using $(4.16 + o(1))\frac{n \log_2 n}{\delta}$ auctions.*

We also study the performance of AuctionIG in the special cases when $m = 1$ and $m = n$, i.e., when there is only one coalition and when each agent is in a separate coalition. The proof is given in Appendix B.5.

Theorem 4.2. *Let m be the number of coalitions.*

(a) When $m = 1$, the sample complexity of AuctionIG is bounded by $2n \log_2 n + 4n$ determinsitically.
*(b) When $m = n$, the **expected** sample complexity of AuctionIG is exactly $nH_n \leq (0.70 + o(1))n \log_2 n$.*

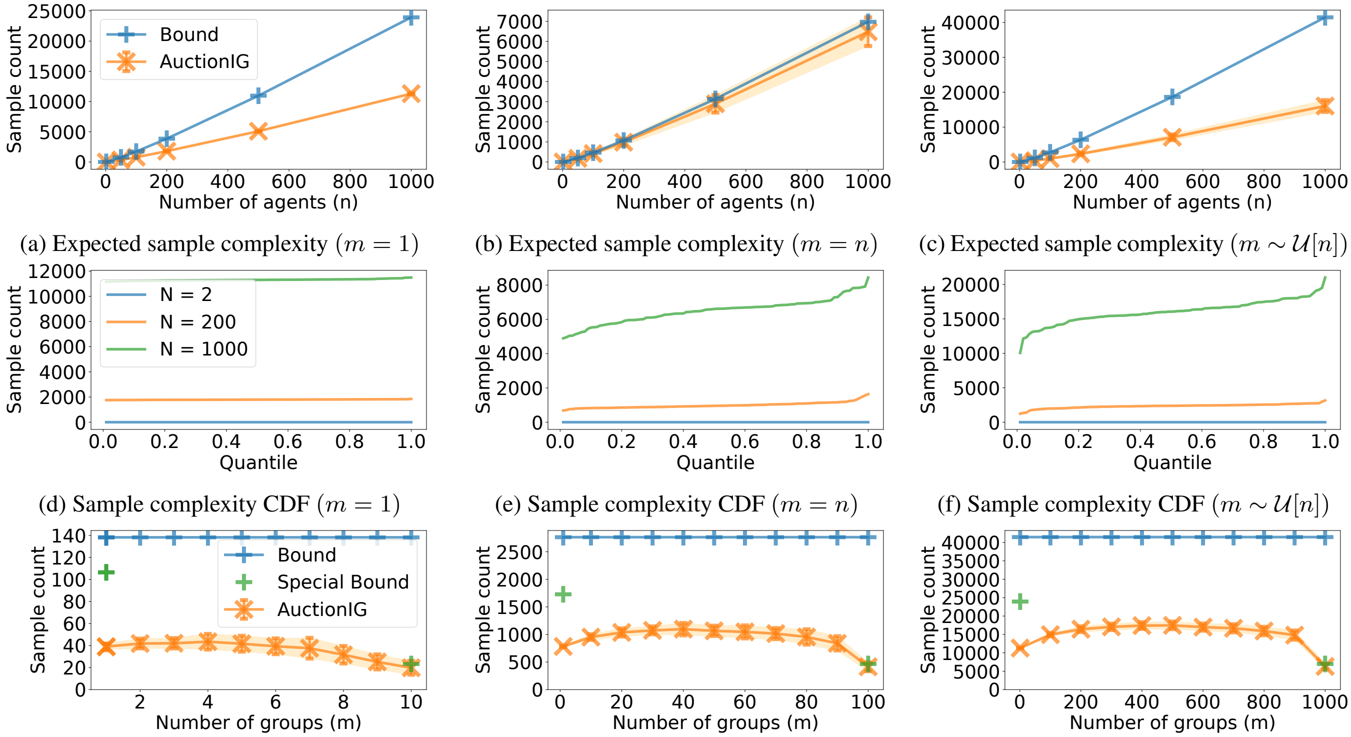
5 Experiments

We conduct experiments to evaluate the performance of our algorithms in practice. As IG (Algorithm 1, normal form games) is deterministic and theoretically optimal (up to low order terms) in sample complexity, we only evaluate AuctionIG (Algorithm 2, auctions). We implement it in Python and evaluate it on a server with 56 cores and 504G RAM, running Ubuntu 20.04.6. The source codes can be found at https://github.com/YixuanEvenXu/coalition-learning.

Experiment setup. We evaluate AuctionIG under different settings of n and m, where n is the number of agents and m is the number of coalitions. For each setting, we fix n and either fix m or sample m from $\mathcal{U}[n]$. Then, we synthesize a coalition structure $\mathcal{S}^*$ with exactly n agents and m coalitions at random. We then run AuctionIG, check the correctness of its output, and record the sample complexity (the total number of samples used). We repeat this process 100 times and report the distribution of the sample complexity. We also report the theoretical upper bound of the expected sample complexity given by Theorem 4.1 and whenever applicable Theorem 4.2. The results are shown in Fig. 4.

AuctionIG's performance with different n. As shown in Figs. 4a to 4c, we let $n = \{2, 50, 100, 200, 500, 1000\}$ and consider fixing $m = 1$ (Fig. 4a), fixing $m = n$ (Fig. 4b) and sampling m from $\mathcal{U}[n]$ (Fig. 4c). For $m = 1, n$, we apply the bounds given in Theorem 4.2, and for $m \sim \mathcal{U}[n]$, we apply the bound given in Theorem 4.1. The results show that the actual performance of AuctionIG is always within a constant factor of its theoretical bounds given in Theorems 4.1 and 4.2. Moreover, when $m = n$, the actual performance is very close to the theoretical bound.

AuctionIG's performance with different m. As shown in Figs. 4g to 4i, we let $m = \{1, 0.1n, 0.2n, \ldots, n\}$ and consider fixing $n = 10$ (Fig. 4g), $n = 100$ (Fig. 4h) and $n = 1000$ (Fig. 4i). We plot the theoretical bounds given in Theorem 4.1 for all m and those given in Theorem 4.2 for $m = 1, n$. The results show that when $m \in (1, n)$, the sample complexities of AuctionIG are similar across different values of m. However, when $m = 1, n$, the sample complexities are significantly lower. This trend is increasingly visible when n grows larger. This shows that Theorem 4.2 complements Theorem 4.1 well in the sense that it provides a tighter bound for the special cases when $m = 1, n$.



(a) Expected sample complexity $(m = 1)$ (b) Expected sample complexity $(m = n)$ (c) Expected sample complexity $(m \sim \mathcal{U}[n])$

(d) Sample complexity CDF $(m = 1)$ (e) Sample complexity CDF $(m = n)$ (f) Sample complexity CDF $(m \sim \mathcal{U}[n])$

(g) Expected sample complexity $(n = 10)$ (h) Expected sample complexity $(n = 100)$ (i) Expected sample complexity $(n = 1000)$

Figure 4: Performance of AuctionIG under different settings of n and m. *Bound* refers to the theoretical bound of its expected sample complexity given by Theorem 4.1 and whenever applicable Theorem 4.2. *AuctionIG* refers to the average sample complexity of AuctionIG over 100 runs with error bars indicating its standard deviation. The results show that the actual performance of AuctionIG is always within a constant factor of its theoretical bounds given in Theorems 4.1 and 4.2.

Correct Probability		50%	90%	99%
Bound in Corollary 4.1		82916	414577	4145767
Algorithm	$m = 1$	11306	11401	11482
	$m = n$	6610	7294	7915
	$m \sim \mathcal{U}[n]$	16055	18009	19510

Table 2: The empirical sample complexity of AuctionIG with 50%, 90% and 99% probability of correctness when $n = 1000$ and the corresponding bounds given in Corollary 4.1. The results show that in practice AuctionIG performs much better than the bounds given in Corollary 4.1.

PAC complexity of AuctionIG. As shown in Figs. 4d to 4f, we evaluate the PAC complexity of AuctionIG by plotting the CDFs of its sample complexity over 100 runs under different settings of n and m. We also highlight several points on the CDFs that correspond to the sample complexity of AuctionIG with 50%, 90%, and 99% probability of correctness when $n = 1000$ in Table 2. We can see that the actual sample complexity of AuctionIG is relatively stable across different runs and is much lower than the theoretical bounds given in Corollary 4.1 when we require a high probability of correctness. This is because Corollary 4.1 is derived using Markov's inequality, which is a very loose bound. In fact, with a finer-grained analysis of the coupon collector's problem, we can improve it using the limit distribution of the coupon collector's problem (see e.g. Papanicolaou and Doumas 2020). However, in that way, we will not be able to write the PAC complexity in a simple closed form.

Summary of experiment results. The experiments show that Theorems 4.1 and 4.2 characterize the expected sample complexity of AuctionIG well with a tight constant, especially when $m = n$ where the bounds are almost perfect. Moreover, the empirical PAC complexity of AuctionIG is much lower than the bounds given in Corollary 4.1, demonstrating its practicality.

6 Conclusion and Discussion

In this paper, we propose and study the Coalition Structure Learning (CSL) and AuctionCSL problems under the one-bit observations. We present a novel Iterative Grouping (IG) algorithm and its counterpart AuctionIG to efficiently tackle these problems, both achieving a sample complexity with asymptotically matching lower bounds. Empirical results demonstrate that these algorithms are indeed sample efficient and useful in practice. Future work includes (i) handling cases where players are aware of and are strategically manipulating our algorithm, (ii) handling bounded rationality, (iii) more general classes of observations, and (iv) admitting equilibrium concepts beyond Nash.

Acknowledgements

This work was supported in part by NSF grant IIS-2046640 (CAREER), IIS-2200410, and Sloan Research Fellowship. Additionally, we thank Davin Choo and Hanrui Zhang for helpful discussions about this work.

References

Athey, S.; and Haile, P. A. 2002. Identification of standard auction models. *Econometrica*, 70(6): 2107–2140.

Balcan, M.-F.; Blum, A.; Haghtalab, N.; and Procaccia, A. D. 2015. Commitment without regrets: Online learning in stackelberg security games. In *Proceedings of the sixteenth ACM conference on economics and computation*, 61–78.

Bonjour, T.; Aggarwal, V.; and Bhargava, B. 2022. Information theoretic approach to detect collusion in multi-agent games. In *Uncertainty in Artificial Intelligence*, 223–232. PMLR.

Braess, D. 1968. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12: 258–268.

De Bruijn, N. G. 1981. *Asymptotic methods in analysis*, volume 4. Courier Corporation.

Dowling, J. 2023. How Uber drivers trigger fake surge price periods when no delays exist. *Drive*.

Erdős, P.; and Rényi, A. 1961. On a classical problem of probability theory. *Magyar Tud. Akad. Mat. Kutató Int. Közl*, 6(1): 215–220.

Feller, W. 1991. *An introduction to probability theory and its applications, Volume 2*, volume 81. John Wiley & Sons.

Geiger, P.; and Straehle, C.-N. 2021. Learning game-theoretic models of multiagent trajectories using implicit layers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4950–4958.

Haghtalab, N.; Fang, F.; Nguyen, T. H.; Sinha, A.; Procaccia, A. D.; and Tambe, M. 2016. Three Strategies to Success: Learning Adversary Models in Security Games. In Kambhampati, S., ed., *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 308–314. IJCAI/AAAI Press.

Hamilton, I. A. 2019. Uber Drivers Are Reportedly Colluding to Trigger 'Surge' Prices Because They Say the Company Is Not Paying Them Enough. *Business Insider*.

Jena, P. K.; Ghosh, S.; and Koley, E. 2021. Design of a coordinated cyber-physical attack in IoT based smart grid under limited intruder accessibility. *International Journal of Critical Infrastructure Protection*, 35: 100484.

Kearns, M.; Littman, M. L.; and Singh, S. 2001. Graphical Models for Game Theory. In *Proceedings of the 17th Conference in Uncertainty in Artificial, Intelligence, 2001*, 253–260.

Kenton, Z.; Kumar, R.; Farquhar, S.; Richens, J.; MacDermott, M.; and Everitt, T. 2023. Discovering agents. *Artificial Intelligence*, 103963.

Kuleshov, V.; and Schrijvers, O. 2015. Inverse game theory: Learning utilities in succinct games. In *Web and Internet Economics: 11th International Conference, WINE 2015, Amsterdam, The Netherlands, December 9-12, 2015, Proceedings 11*, 413–427. Springer.

Lakshminarayana, S.; Belmega, E. V.; and Poor, H. V. 2019. Moving-target defense for detecting coordinated cyber-physical attacks in power grids. In *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 1–7. IEEE.

Letchford, J.; Conitzer, V.; and Munagala, K. 2009. Learning and approximating the optimal strategy to commit to. In *Algorithmic Game Theory: Second International Symposium, SAGT 2009, Paphos, Cyprus, October 18-20, 2009. Proceedings 2*, 250–262. Springer.

Ling, C. K.; Fang, F.; and Kolter, J. Z. 2018. What Game Are We Playing? End-to-end Learning in Normal and Extensive Form Games. In Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 396–402. ijcai.org.

Mazrooei, P.; Archibald, C.; and Bowling, M. 2013. Automating collusion detection in sequential games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, 675–682.

Milgrom, P. R. 2004. *Putting auction theory to work*. Cambridge University Press.

Newman, D. J. 1960. The double dixie cup problem. *The American Mathematical Monthly*, 67(1): 58–61.

Paes Leme, R.; Pal, M.; and Vassilvitskii, S. 2016. A field guide to personalized reserve prices. In *Proceedings of the 25th international conference on world wide web*, 1093–1102.

Papanicolaou, V. G.; and Doumas, A. V. 2020. On an Old Question of Erdős and Rényi Arising in the Delay Analysis of Broadcast Channels. *arXiv preprint arXiv:2003.13132*.

Peng, B.; Shen, W.; Tang, P.; and Zuo, S. 2019. Learning Optimal Strategies to Commit To. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 2149–2156.

Ribeiro, C. C.; Urrutia, S.; and de Werra, D. 2023. A tutorial on graph models for scheduling round-robin sports tournaments. *International Transactions in Operational Research*.

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

Sweeney, S. 2019. Uber, Lyft drivers manipulate fares at Reagan National causing artificial price surges. *WJLA*.

Tripathy, M.; Bai, J.; and Heese, H. S. S. 2022. Driver collusion in ride-hailing platforms. *Decision Sciences*.

Waugh, K.; Ziebart, B. D.; and Bagnell, D. 2011. Computational Rationalization: The Inverse Equilibrium Problem. In Getoor, L.; and Scheffer, T., eds., *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, 1169–1176. Omnipress.

Wellman, M. P. 2006. Methods for empirical game-theoretic analysis. In *AAAI*, volume 980, 1552–1556.