

Distributed MIS Algorithms for Rational Agents using Games

Nithin Salevemula and Shreyas Pai

Indian Institute of Technology Madras*

November 21, 2025

Abstract

We study the problem of computing a Maximal Independent Set (MIS) in distributed networks, where each node is a rational agent that receives a payoff depending on whether it is included in the MIS. In classical distributed computing, it is typically assumed that nodes follow the prescribed algorithm faithfully. However, this assumption fails when nodes are rational agents whose utilities depend on the algorithm’s output. In such cases, nodes may deviate from the algorithm if it increases their expected payoff.

Classical solutions for MIS assume that nodes generate random bits honestly or rely on unique identifiers to break symmetry. However, in rational settings, nodes may manipulate randomness to gain a strategic advantage, and relying solely on unique identifiers can result in unfairness, where some nodes have zero probability of joining the MIS and thus no incentive to participate. To address these challenges, we propose two algorithms that work under a utility model, where agents are incentivized to compute locally correct solutions while also exhibiting preferences among these solutions. In these algorithms, randomness is generated through interactions between neighboring nodes, which can be viewed as simple games, where no single node can unilaterally change the outcome. This approach allows us to break symmetry while being compatible with rational behavior.

For both algorithms, we show that regardless of the execution history that has occurred, no agent can improve its expected utility by deviating from that stage, provided no other agents deviate. This is a much stronger guarantee compared to Trembling Hand Perfect Equilibrium, which is typically used in such scenarios. Both algorithms guarantee that when all the nodes follow the algorithm, every node has a positive probability of joining the MIS, and that the final output is a correct Maximal Independent Set. Finally, for both algorithms, we can guarantee termination in $O(\log n)$ rounds with high probability under mild additional assumptions, where n is the number of nodes in the network.

1 Introduction

In many classical distributed computing settings, some fraction of the nodes is assumed to be byzantine while the rest are obedient. An obedient node follows the prescribed algorithm without deviation, whereas byzantine nodes are assumed to be controlled by an adversary that is actively trying to ensure that the algorithm’s goals are not achieved. Such a setup is valid as long as the obedient nodes follow the prescribed algorithm perfectly and the fraction of byzantine nodes is not too high. However, in many real-world scenarios, the situation is a bit more nuanced: nodes in the network often have their own incentives, which makes algorithm design challenging even without the presence of a byzantine adversary. In this work, we take a ”middle path” of assuming that each node in the network is a selfish agent with its own utility function that it wants to maximize. If we allow for arbitrary utility functions, the behavior of the rational nodes will be

*Email: nithinprakash9999@gmail.com, shreyas@cse.iitm.ac.in

similar to that of a byzantine node¹. Therefore, we restrict our attention to utility functions where nodes are incentivized to compute a correct solution, but may still have a preference over which correct solution they want to compute. See Section 2 for a full description of our model.

Consider the Maximal Independent Set (MIS) problem as a working example. Although all nodes are primarily interested in ensuring that the MIS is computed correctly, they also prefer to be included in the set. As a result, a node may deviate from the prescribed algorithm if doing so increases its chance of joining the MIS. This behavior is referred to as *rationality* in the game-theoretic literature, where agents (nodes) act to maximize their individual utility. Deviations due to individual rationality can harm both the correctness and termination guarantees of classical MIS algorithms. Therefore, we need to design new algorithms that are resilient to rational behavior. Such algorithms guarantee that nodes, acting in their own self-interest, still prefer to follow the algorithm rather than deviate.

Challenges for Computing MIS with Rational Agents Classical distributed algorithms for computing a Maximal Independent Set (MIS), such as Luby’s algorithm [13] and the algorithm by Métivier et al. [15], are not suitable in rational settings where nodes may deviate in order to maximize their own utility. As an illustrative example, let us examine both these algorithms under the utility model where nodes are incentivized to be included in the MIS.

In Luby’s algorithm [13], each node i proposes to join the MIS with probability $1/2d(i)$, where $d(i)$ is the degree of the node i . A node successfully joins the MIS if none of its neighbors propose in the same round. However, this approach fails in a rational setting: since the decision to propose is determined by internal randomness, a rational node can always falsely claim that it has proposed to join the MIS with probability $1/2d(i)$, but it got lucky in every round. If all nodes act this way, no node ever joins the MIS, and the algorithm never terminates. In the algorithm of Métivier et al. [15], each node generates a rank uniformly at random from the interval $[0, 1]$. A node joins the MIS if its rank is lower than all of its neighbors’ ranks. Again, this algorithm will not work if nodes prefer joining the MIS: a rational node can simply report a rank of 0 in every round, ensuring it always appears eligible to join. If all nodes behave similarly, no node is eliminated, no progress is made, and the algorithm stalls indefinitely.

Given the above discussion, one may argue that we should then just focus on deterministic MIS algorithms, as there has been impressive progress on this front [8]. However, deterministic algorithms have the following problem: a node may realize that it can never join the MIS by following the algorithm, even though none of its neighbors have currently joined the MIS. In such a situation, it will have no incentive to participate further in the algorithm. As a simple example, consider a deterministic algorithm where, in each round, a node joins the MIS if its ID is smaller than those of all its neighbors. If a node i has a neighbor j with the globally minimum ID, i knows it can never join the MIS even though j has not yet joined the MIS. Therefore, i has no incentive to participate in the algorithm. In fact, any deterministic algorithm would face the same problem when nodes are rational agents aiming to maximize their own utility. This problem even holds for algorithms that have a randomized part followed by a deterministic part.

These examples demonstrate that classical MIS algorithms do not translate directly to settings with rational agents. Therefore, new algorithms must be designed that are resilient to deviations that arise from rational behavior and guarantee participation from every node.

Our Contributions We use the classical framework of *extensive-form games* to analyze the MIS problem in distributed networks with rational agents. Each node in the network is modeled as a rational player, motivated by the incentive to be included in the MIS. We define a *strategy algorithm* that specifies, for each node, the action it should take in every round until termination, as a function of its state—including those that may arise due to deviations from the prescribed algorithm, either by itself or by other nodes. We then propose two such strategy algorithms

¹Albeit in this case, all the byzantine nodes may not be controlled by a single adversary

in which nodes have no unilateral incentive to deviate at any state, even at states that occur as a result of earlier deviations. We refer to this property as a *belief-independent sequential equilibrium*, which provides a stronger guarantee than Nash equilibrium, sequential equilibrium, or trembling-hand perfect equilibrium in games of imperfect information.

Furthermore, when all agents follow the strategy algorithm, the nodes joining the independent set form a valid MIS, and every node has a non-zero probability of being included in the MIS. The following is a brief description of our proposed strategy algorithms.

- The first algorithm (Section 3): In each iteration, every active node plays a simple two-player game of rock-paper-scissors with each of its neighbors in order to decide who gets to join the MIS. Only a node that wins all its games against its neighbors in an iteration gets to join the MIS. This rule allows us to ensure that no agent can benefit from unilaterally deviating from the algorithm at any point. Moreover, when all nodes follow the algorithm, we can guarantee termination in $O(\log n)$ rounds with high probability, assuming the maximum degree of the network graph is constant, where n is the number of nodes in the network.
- The second algorithm (Section 4) uses lightweight cryptographic assumptions to generate randomness in a way that prevents manipulation. Each agent computes its priority for the current round by combining its own random value with a signed random value received from one of its neighbors. Since the randomness is jointly determined, no agent can unilaterally bias the outcome. These priorities are then used to simulate one round of the algorithm of Métivier et al. [15]. Repeating this process, the algorithm guarantees termination in $O(\log n)$ rounds with high probability.

2 The LOCAL Model with Rational Agents

We work in the LOCAL model, a classic synchronous message-passing model of distributed computing [12]. The network is abstracted as $G = (V, E)$, an undirected graph, where V is the set of nodes (agents) and $E \subseteq V \times V$ is the set of edges. As usual we denote $n = |V|$ and $m = |E|$. Each node represents a selfish agent that acts to maximize its own utility, and each edge $(i, j) \in E$ indicates that agents i and j can communicate directly. We assume no prior communication occurs between the agents before the algorithm begins. Each agent is expected to irrevocably output one of three values: 1, declaring itself as a member of the MIS; 0, declaring itself to not belong to the MIS; or \perp , indicating an abort. The output of a node, once computed, is visible to all the neighbors of that node.

The utility of each node depends only on its own output and those of its neighbors. As discussed earlier, if we allow for arbitrary utilities, the behavior of the rational nodes will be similar to that of Byzantine nodes, which will lead to trivial impossibilities. Therefore, we consider a particular family of utility functions in which nodes are incentivized to compute a locally correct solution. Additionally, each node i gets utility value $v_i > 0$ if it outputs 1 and all its neighbors output 0. More precisely, the utility function for a node i can be defined as a function on its own output and its neighbors' output. Let $\text{out}(i) \in \{1, 0, \perp\}$ denote the output of node i and let $\text{out}(N(i))$ denote the list of outputs of all its neighbors. Then, the utility function u_i for node i is defined in Equation 1. Here we slightly abuse notation by interpreting $\text{out}(N(i))$ as a set in the case analysis. The conditions are evaluated in order; once a condition is satisfied, the subsequent conditions are not considered, even though they may be applicable.

Given the above model and assumptions, our goal is to design a distributed algorithm in which nodes, despite being able to deviate, find it optimal to follow the algorithm. However, if a deviation does occur, we cannot simply ignore the deviating node; the algorithm must still prescribe how it should optimally behave from that point onward. In other words, the algorithm should tell each node what to do in every situation it could possibly face, even those that would

$$u_i(\text{out}(i), \text{out}(N(i))) = \begin{cases} 0, & \text{if } \text{out}(i) = \perp \text{ or } \perp \in \text{out}(N(i)) \quad (\text{locally aborted}), \\ 0, & \text{if } \text{out}(i) = 0 \text{ and } 1 \in \text{out}(N(i)) \quad (\text{locally valid, } i \text{ not in MIS}), \\ v_i, & \text{if } \text{out}(i) = 1 \text{ and } 1, \perp \notin \text{out}(N(i)) \quad (\text{locally valid, } i \text{ in MIS}), \\ -\infty, & \text{otherwise} \quad (\text{locally invalid solution}). \end{cases}$$

Equation 1: Local utility function for node i .

occur only if some of them deviated earlier. This is precisely analogous to the notion of a *strategy* in game theory. To capture this requirement, we introduce the notion of a *strategy algorithm*. Intuitively, it specifies how a node should act in every possible situation it may encounter, including those that arise due to deviations by itself or its neighbors. To define this formally, we first define the notion of a *state*, and then use it to precisely characterize what constitutes a strategy algorithm.

Definition 2.1 (State). *The state of a node i at the beginning of round t consists of all messages sent and received by i in rounds $t' < t$, the values of its local variables (including any private randomness), and any observable outputs of its neighbors from earlier rounds.*

Definition 2.2 (Strategy Algorithm). *A strategy algorithm in the LOCAL model with rational nodes is a special type of algorithm, which specifies, for each node i , the action it should take in every round until termination as a function of its state, including states that may arise due to deviations from the prescribed algorithm by itself or by other nodes.*

Remark 2.3. *This means that in every round, all nodes know the precise internal computation and the external action prescribed by the algorithm. However, whether a player actually follows the prescribed action is left to the node's discretion. We assume that a player cannot deviate in its internal deterministic computations; thus, the only possible deviations arise from (1) internal computations involving randomness or (2) external actions such as sending incorrect messages, omitting required messages, outputting values at unintended times, or outputting values different from those prescribed by the algorithm. This assumption is without loss of generality, since any deviation in deterministic internal computation must eventually manifest in the node's external behavior to have any observable effect. Therefore, it suffices to model deviations as occurring in the external actions of a node rather than in its internal deterministic computation.*

Intuition for our utility function and modeling choices Our aim in this paper is to study rational behavior in the LOCAL model. A basic feature in many LOCAL model algorithms is that the nodes perform computation for a certain number of rounds and finally produce a correct output upon termination. When the nodes are rational agents, we want them to terminate at some point; therefore, it is essential that once a node commits to an output, it cannot be changed. Moreover, our utility function is well-defined only once a node and its neighbors have computed their outputs, and allowing nodes to change their outputs makes it difficult to calculate their utility. We can still design algorithms where nodes are allowed to change their output by having nodes compute a temporary output that can be changed. But before termination, the nodes have to irrevocably commit to this output, and the utility is calculated based on the committed outputs of all nodes.

Next, we motivate our choice of utility function. If all nodes are in a locally valid solution, we have computed a (globally) correct MIS, and since we are interested in computing a correct MIS solution, we require that all nodes prefer a locally valid solution over a locally invalid solution. In fact, we do not want the nodes to try to join the MIS at a risk of being in a locally invalid

solution with some probability. Hence, we assign a utility of $-\infty$ to nodes in a locally invalid solution. Moreover, MIS has the property that if a node is in a locally invalid solution, it can always change its own output to be in a locally valid solution. Since an output change is not allowed, this means that a node in a locally invalid solution has committed too early to its output, and such behavior must be appropriately penalized.

If we have the same utility for all locally valid solutions, we are in the standard LOCAL model, as the rational nodes will faithfully follow any algorithm that computes a correct MIS. Therefore, to make the model interesting, we need to assign different utility values to different locally valid solutions. There are indeed many choices here, but for the purposes of this paper, we study one family of utility functions where nodes always prefer locally valid solutions where they belong to the MIS. The other choices in this space are also interesting, especially the case where nodes prefer locally valid solutions where they are not in the MIS. Designing rationality-resilient algorithms for these alternative families of utility functions is left as an open problem.

We now address why we introduce the third output symbol, \perp . If we define the utility function with only two output labels 0 and 1, we may encounter the following scenario: if a node outputs 1 by deviating from the protocol to get higher utility, then the best response for all its neighbors is to output 0, since otherwise they obtain $-\infty$ utility. Therefore, the node that outputs 1 is getting away with its deviation simply because its neighbors can do nothing but support its action. To address this problem, we need a way for nodes to retaliate when their neighbors deviate from the algorithm and irrevocably output 1. We do this by introducing a new output symbol \perp which can be interpreted as a credible threat. Now, if a node i deviates by outputting 1, its neighbors can output \perp instead of 0 at no extra penalty to themselves, which makes the deviation of i not profitable. It is important to note that the output \perp is only used as a credible threat to ensure nodes do not deviate from the algorithm. In a correct execution of the algorithm with no deviations, no node will ever actually output \perp . So a node outputting \perp can be considered as a failure of the algorithm execution.

We also assume that the output of each node is visible to all of its neighbors. This assumption is necessary because, in our model, the utilities of nodes are based on their own output and the outputs of their neighbors. If outputs of neighbors are not directly observable, a node would have to rely solely on what its neighbors report as their output, which introduces the possibility of misreporting. In such a case, a trivial but undesirable outcome could arise where all nodes output 1 while falsely reporting to have output 0 to their neighbors, resulting in a configuration that is not a valid MIS, as every node is part of the MIS in reality. Hence, visibility of the neighbor's output is required to ensure that utilities are well-defined and correspond to actual local configurations.

2.1 MIS with Rational Nodes as Extensive-Form Game

As discussed in Section 2, we consider a model in which nodes are *rational* and interact with one another to change their states. Each node's utility depends on the final state of the network, making the overall system naturally analogous to a strategic game among the nodes. Since these notions are well formalized in the game-theoretic framework, we model our setting as an *extensive-form game*.

In an r -round LOCAL model algorithm, a node has no information about the graph outside its r -hop neighborhood. This means the corresponding extensive form game must be one with *imperfect information*. Since each node retains complete memory of all its past actions and observations, the game satisfies the property of *perfect recall*. Finally, because we assume synchronous communication rounds in which all nodes act simultaneously, the game can be viewed as an *extensive-form game with imperfect information, perfect recall, and simultaneous moves*.

Formally, an extensive-form game is defined as a tuple consisting of several components (see [7] for a complete description). In our context, we do not require the full generality of this definition; it suffices to specify only the elements necessary to model our distributed setting.

Accordingly, we work with a simplified version of the tuple and its associated definitions.

Definition 2.4 (Distributed MIS Problem with Rational Agents as an Extensive Form Game with Imperfect Information, Perfect Recall and Simultaneous Moves). ² Given a graph $G = (V, E)$, where V is the set of nodes (agents) and $E \subseteq V \times V$ is the set of edges and each edge $(i, j) \in E$ indicates that agents i and j can communicate directly. The MIS problem over G can be modeled as an extensive game, denoted by \mathcal{G} . Formally, we define \mathcal{G} as a tuple $\mathcal{G} = \langle V, H, P, \{\mathcal{I}_i\}_{i \in V}, \{A_i\}_{i \in V}, \{\succsim_i\}_{i \in V} \rangle$ whose components are described below.

- V : the set of players, corresponding to the nodes of the graph.
- H : the set of all possible prefix-closed histories of messages exchanged between neighboring nodes and their outputs, subject to the following constraints: each node produces an output at most once, and once a node outputs, it becomes inactive—i.e., it can no longer send or receive messages.
- A history $h \in H$ is called terminal if it is infinite, or if all nodes output a value from $\{0, 1, \perp\}$. Denote by Z the set of all terminal histories.
- $P : H \setminus Z \rightarrow 2^V \setminus \{\emptyset\}$: assigns to each nonterminal history h the set of players who move simultaneously at that history. $P(h)$ is the set of nodes that have not yet produced an output at history h .
- Since any player $i \in V$ observes only the messages it has sent and received, together with the outputs of its neighbors, so every history $h \in H$ has a local projection onto the information accessible to i . Formally, we define the projection function $\text{proj}_i : H \rightarrow H_i$, where H_i denotes the set of all possible local histories of player i . For any global history $h \in H$, the projection $\text{proj}_i(h)$ is the subsequence of events in h that involve i —namely, all messages sent or received by i , together with the outputs of its neighbors observed by i . Two global histories $h, h' \in H$ are said to be indistinguishable to i if they induce the same local history, that is, $h \sim_i h' \iff \text{proj}_i(h) = \text{proj}_i(h')$. This indistinguishability relation \sim_i partitions the set $\{h \in H \mid i \in P(h)\}$ of global histories where i is active into equivalence classes. We denote this partition by \mathcal{I}_i , where each element $I_i \in \mathcal{I}_i$ corresponds to the set of all global histories that induce the same local history to i . Each such class I_i is therefore an information set of player i , since player i cannot distinguish between any two global histories in the same class and therefore must choose the same action at all of them. Let $I_i(h_i)$ denote the set of all global histories that project local history h_i on i .
- At any information set $I_i \in \mathcal{I}_i$. The set of possible actions for player i , denoted $A_i(I_i)$, consists of any combination of: (1) perform internal computation, (2) send arbitrarily long messages to any subset of undecided neighbors, and (3) output a value in $\{0, 1, \perp\}$.
- For each player $i \in V$, a preference relation \succsim_i is defined over terminal histories Z , representable by a local utility function $u_i : Z \rightarrow \mathbb{R}$. For an infinite history $h \in Z$, the utility of all nodes is defined to be 0. For a finite terminal history, the utility is given by the function in Equation 1.

Given the definition of the game, we can define pure, mixed, and behavioral strategies as follows:

Definition 2.5 (Pure, Mixed and Behavioral Strategies [17, Definitions 203.1, 212.1]). A pure strategy of player $i \in V$ of an extensive-form game \mathcal{G} is a function that assigns an action in $A_i(I_i)$ to each information set $I_i \in \mathcal{I}_i$. A mixed strategy of player i is a probability measure over

²This is an instance of an extensive form game, refer [7, Chapter 12] for the exact definitions of extensive form game with imperfect information, perfect recall and simultaneous moves.

the set of i 's pure strategies. Whereas a behavioral strategy of player i is a collection $(\beta_i(I_i))_{I_i \in \mathcal{I}_i}$ of independent probability measures, where $\beta_i(I_i)$ is a probability measure over $A(I_i)$.

Remark 2.6. Under this formulation, a strategy algorithm can be viewed as a behavioral strategy, since it specifies, for each node, the action (or distribution over actions) it should take at every possible state it may encounter, which corresponds to the local history in the extensive form game \mathcal{G} defined above.

Next, we define various equilibrium notions for extensive-form games before introducing the equilibrium guarantee that can be obtained.

Definition 2.7 (Nash equilibrium [17, Chapter 11.5]). A Nash equilibrium in mixed strategies of an extensive game is a profile $\sigma^* = (\sigma_i^*)_{i \in N}$ of mixed strategies with the property that for every player $i \in N$ we have $O(\sigma_{-i}^*, \sigma_i^*) \succeq_i O(\sigma_{-i}^*, \sigma_i)$ for every mixed strategy σ_i of player i , where $O(\sigma)$ denotes the expected outcome (or distribution over terminal histories) induced by the strategy profile σ and σ_{-i}^* denotes the equilibrium strategy profile of all players except i .

Definition 2.8 (Sequential Equilibrium [17, Definition 222.1]). A sequential equilibrium of an extensive-form game \mathcal{G} is an assessment (β, μ) , where $\beta = (\beta_i)_{i \in N}$ is a profile of behavioral strategies (which assign, for each information set $I_i \in \mathcal{I}_i$, a probability distribution over the available actions at I_i) and μ is a belief system, that is, a function that assigns to every player i and every information set $I_i \in \mathcal{I}_i$ a probability measure over the set of histories contained in I_i , conditional on I_i being reached. The assessment (β, μ) is a sequential equilibrium if it satisfies the following conditions:

1. **Sequential Rationality:** An assessment (β, μ) is said to be sequentially rational if, for every player $i \in V$ and every information set $I_i \in \mathcal{I}_i$, the strategy β_i prescribes actions that maximize i 's expected utility at I_i , given the belief system μ and the strategies β_{-i} of the other players. Formally, let $O(\beta, \mu \mid I_i)$ denote the distribution over terminal histories induced by (β, μ) conditional on I_i being reached. Then (β, μ) is sequentially rational if $O(\beta, \mu \mid I_i) \succeq_i O((\beta_{-i}, \beta'_i), \mu \mid I_i)$ for every behavioral strategy β'_i of player i .
2. **Consistency:** An assessment (β, μ) is said to be consistent if there exists a sequence of assessments $\{(\beta^n, \mu^n)\}_{n=1}^\infty$ that converges to (β, μ) in Euclidean space, where each strategy profile β^n is completely mixed and each belief system μ^n is derived from β^n using Bayes' rule.

Another equilibrium notion that is considered is called *Trembling Hand Perfect Equilibrium*, which is a refinement of sequential equilibrium with a particular assessment. It ensures that each player's strategy remains optimal, even if players occasionally make unintended moves with a very small probability, thereby ensuring that strategies are robust to such "trembles" in decision-making. While both sequential and trembling hand perfect equilibria provide a guarantee only with respect to a particular belief system (that is, a strategy is optimal given that belief system), Our strategy algorithms give behavioral strategies that admit a stronger guarantee: they are Sequentially Rational for *any* belief system. Hence, unlike sequential equilibrium, we do not define behavioral strategies relative to a belief system, But only over information sets. We formalize this stronger notion as a Belief-Independent Sequential Equilibrium.

Definition 2.9 (Belief-Independent Sequential Equilibrium). A behavioral strategy profile $\beta^* = (\beta_i^*)_{i \in N}$ of an extensive-form game \mathcal{G} is an Belief-Independent Sequential Equilibrium if, for every player i and every information set $I_i \in \mathcal{I}_i$, the behavioral strategy β_i^* is sequentially rational, regardless of player i 's belief over the histories in I_i . Formally, for every player $i \in V$, every information set $I_i \in \mathcal{I}_i$, any belief system μ of player i , $O(\beta^*, \mu \mid I_i) \succeq_i O((\beta_{-i}^*, \beta'_i), \mu \mid I_i)$ for every behavioral strategy β'_i of player i . where β_{-i}^* denotes the equilibrium strategies of all players other than i in the profile β^* .

Now we are ready to define the desirable properties of a strategy algorithm.

Definition 2.10 (Rationality Resilient Strategy Algorithm). *A Strategy Algorithm is said to be rationality resilient (with respect to the above MIS problem) if it satisfies the following properties*

1. **Belief-Independent Sequential Equilibrium:** *The behavioral strategy corresponding to the strategy algorithm should induce Belief-Independent Sequential Equilibrium 2.9.*
2. **Termination:** *If all nodes follow the algorithm, it must terminate after a finite number of rounds with high probability.*
3. **Correctness:** *If all nodes follow the algorithm and terminate, the set of nodes that output 1 forms a valid Maximal Independent Set (MIS), and all remaining nodes output 0.*
4. **Positive Inclusion Probability:** *If all nodes follow the algorithm, then each node must have a non-zero probability of eventually outputting 1 in every round where none of its neighbors has output 1. This ensures that no node prematurely drops out of participation based on the belief that it has no future chance of joining the MIS.*

2.2 Related Work

The study of incentive-compatible distributed algorithms has led to the field of *Distributed Algorithmic Mechanism Design (DAMD)* [5], which extends the classical *Algorithmic Mechanism Design (AMD)* framework to distributed settings where there is no trusted central authority. While AMD focuses on designing truthful mechanisms in centralized settings, DAMD addresses the challenge of ensuring that rational agents follow prescribed strategies when the computation is decentralized.

Nisan [16] gave a distributed mechanism for computing a truthful maximum independent set in chain networks. Later, Hirvonen and Ranjbaran [11] proposed a distributed mechanism for all local optimization problems, including Maximal Independent Set, that guarantees a Δ -approximation. However, both approaches require monetary transfers (payments) to third parties to enforce truthfulness – something we explicitly aim to avoid in our work. Moving beyond the third-party setting, other widely studied notions include *self-stabilization* [18] [19] and *fairness* [6] [2]. The work by Amoussou-Gueno et al. [3] analyzes the dynamics between rational and Byzantine players in blockchain consensus protocols, highlighting their implications for security and efficiency in decentralized systems.

Rationality has also been extensively studied in the context of the *Secret Sharing* problem. Halpern et al. [10] introduced the notion of rational secret sharing, showing that classical schemes fail when agents act in a selfish manner, and proposed randomized protocols that restore incentive compatibility. Subsequent works [1, 9, 14] explored alternative utility models and provided further constructions under these variations. Although the problem domain is different, these works share with ours the high-level goal of designing distributed algorithms that are resilient to rational deviations.

Solving *Locally Checkable Labeling (LCL) problems* with rational nodes was formalized as extensive-form games by Collet et al. [4]. They show that for all LCL problems solvable by greedy sequential algorithms – such as MIS, $(\Delta + 1)$ -coloring, and maximal matching – there exist distributed algorithms that are robust to selfish behavior and form trembling hand perfect equilibria. While we adopt an equivalent extensive-form game framework, our model differs in several key respects. First, in their setting, actions are restricted to choosing labels, whereas in our model, nodes may also send arbitrary messages to their neighbors. Second, their formulation allows labels to be revised in later stages, while in our setting, label choices, once made, cannot be changed. Finally, in their model, the game always terminates in a valid LCL solution, whereas in our framework, the outcome may correspond to an invalid solution. In our work, we study the MIS problem under a specific family of utility functions and present a strategy that is

sequentially rational irrespective of beliefs, which is a much stronger guarantee than trembling hand perfect equilibrium.

Fineman et al. [6] define fairness in the context of computing maximal independent sets as balanced inclusion probabilities across nodes. However, they do not allow nodes to have rational behavior, and hence their algorithm is not robust to strategic deviations. In contrast, Abraham et al. [2] study fair leader election for specific topologies such as chains and cliques. Their model assumes that agents strictly prefer the existence of some leader over no leader, which is conceptually similar to our assumption that agents prefer to be included in the MIS. Importantly, their protocols are also resistant to collusion. Our algorithms adopt a similar idea of generating randomness from inputs provided by multiple agents with conflicting preferences over the outcome.

3 A Strategy Algorithm for MIS with Rational Nodes

We now describe our first strategy algorithm, which employs a simple and symmetric tie-breaking mechanism: each node plays a Rock-Paper-Scissors (RPS) game with each of its neighbors, and only the nodes that win with all their neighbors are allowed to join the MIS. We note that the RPS game between two nodes i and j must be well defined even if one or both players choose not to play the game. If both nodes send valid moves to each other, then the outcome is determined by the standard RPS rules: rock breaks scissors, scissors cut paper, paper covers rock. If exactly one of the two nodes sends an invalid move or sends no move at all, then the other player wins. If both nodes fail to send valid moves, the result is a tie. We capture all these cases formally in the following definition.

Definition 3.1 (RPS Outcome Function). *Let $rps_{i,j} : \{r, p, s, \phi\}^2 \rightarrow \{i, j, tie\}$ be a function which determines the winner between two nodes i and j that are playing an RPS game with each other. Here r , p , and s correspond to rock, paper, and scissors, respectively, and ϕ denotes an invalid or missing move (i.e., any input not in $\{r, p, s\}$). When node i plays move s_i and node j plays move s_j , the function is defined as follows:*

$$rps_{i,j}(s_i, s_j) = \begin{cases} i, & \text{if } (s_i, s_j) \in \{(r, s), (p, r), (s, p), (r, \phi), (p, \phi), (s, \phi)\} \\ j, & \text{if } (s_i, s_j) \in \{(s, r), (r, p), (p, s), (\phi, r), (\phi, p), (\phi, s)\} \\ tie, & \text{if } s_i = s_j \end{cases}$$

RPS Algorithm Overview The algorithm proceeds in *iterations*, and each iteration consists of *three rounds*. Initially, all nodes are in an **undecided** state. Once a node outputs a value in $\{0, 1, \perp\}$, it becomes **decided** and takes no further action. Each node maintains a **BeenCheated** flag, initialized to **false**, and in every iteration, the strategy algorithm proceeds through the following rounds:

- **Round 1 (Play RPS):** Each node i selects a move from $\{r, p, s\}$ uniformly at random for each of its undecided neighbors and sends it. The RPS games with different neighbors are treated independently, allowing the node to play distinct moves against each of its neighbors.
- **Round 2 (Join MIS if possible):** Each node i checks whether any neighbors has already output 1 or \perp . If such neighbors exist, they are considered to have deviated from the algorithm, since they were not supposed to do so under the prescribed algorithm. In that case, i ignores the rest of the algorithm and outputs \perp in every subsequent round where it remains **undecided**. Otherwise, if no neighbor has output 1 or \perp , then if i wins against all its **undecided** neighbors according to Definition 3.1, it outputs 1 and joins the

MIS. In addition, if i did not receive a move from an **undecided** neighbor in the previous round, it sets its **BeenCheated** flag to **true**.

- **Round 3 (Respond to neighbor outputs):** Each node i checks whether any neighbor j has output \perp . If such a node exists, i outputs \perp in every subsequent round in which it remains **Undecided**.

If any neighbor j has output 1 in previous round, then:

- If i lost to all such neighbors then node i outputs 0 in every subsequent round while it remains undecided, unless **BeenCheated** is **true**, in which case it outputs \perp in every subsequent round while it remains undecided.
- Otherwise, if there exists a neighbor k that lost to i but still output 1 then i outputs \perp in every subsequent round while it remains undecided.

Node i outputs 1 if it has no undecided neighbors; otherwise, it proceeds to the next iteration.

3.1 Rationality Resilient RPS Algorithm

The distributed strategy algorithm described above is formalized in Algorithm 1. This is followed by an analysis showing that this strategy is rationality resilient, encapsulated in Theorem 3.2.

Theorem 3.2. *Algorithm 1 is a Rationality Resilient Algorithm as defined in Definition 2.10.*

We prove Theorem 3.2 in four parts (Theorems 3.5, 3.6, 3.7, and 3.8). Among these, Theorem 3.5 requires intermediate results, which we establish through Lemmas 3.3 and 3.4.

Lemma 3.3. *Let the game be at an arbitrary history h with $i \in P(h)$, where $h \in I_i \in \mathcal{I}_i$. Suppose that one of i 's **UndecidedNeighbors** (say, j) has its **BeenCheated** flag set to **true**. Then, regardless of the history h and of i 's strategy from h onward, if all the other nodes follow Algorithm 1 from h onward, the utility of node i will be at most 0.*

Proof. According to Algorithm 1, node j outputs 0 only in Round 3 of an iteration, and even then, only if its **BeenCheated** flag is **false**. Since the **BeenCheated** flag, once set to **true**, is never reset to **false**, a node with **BeenCheated** = **true** will never output 0. Therefore, node j will only output 1 or \perp . In either case, the utility of all its neighbors, including i , is at most 0. \square

Lemma 3.4. *Let the game be at an arbitrary history h with $i \in P(h)$, where $h \in I_i \in \mathcal{I}_i$, and let h_i be the local history of node i . Then, provided that i and all other undecided nodes follow Algorithm 1 from this point onward, the expected utility of node i is at least 0.*

Proof. A node i receives a utility less than zero (i.e., $-\infty$) only in two cases:

1. i outputs 0, and none of its neighbors output 1, or
2. i outputs 1, and at least one of its neighbors also outputs 1.

Case 1 is not possible under the Algorithm 1: a node is instructed to output 0 only in Round 3 of an iteration, and only after observing at least one neighbor output 1. Hence, after any local history where the node i is undecided in Round 3 outputs 0, it must be the case that a neighbor has already output 1. Therefore, the first case cannot occur.

Case 2 is also prevented by Algorithm 1: a node outputs 1 only in Round 2 or Round 3 of an iteration.

Algorithm 1: RPS based MIS Strategy algorithm for node i

```
1 Initialization
2   UndecidedNeighbors  $\leftarrow N(i)$ ;
3   BeenCheated  $\leftarrow$  false;
4 if  $N(i) = \emptyset$  then
5   | From this point onward, disregard the rest of the algorithm and Output 1 in every
   |   round where node  $i$  is still UNDECIDED;
6 while  $i$  is UNDECIDED do
7   | /* Round 1: Play RPS with Undecided Neighbors */
8   | foreach  $j \in \text{UndecidedNeighbors}$  do
9   |   | Choose strategy  $s_i(j) \in \{r, p, s\}$  uniformly at random and send to  $j$ ;
10  | /* Round 2: Join MIS if possible */
11  | foreach  $j \in \text{UndecidedNeighbors}$  do
12  |   | if  $j$  has output 1 or  $\perp$  then
13  |   |   | From this point onward, disregard the rest of the algorithm and Output  $\perp$  in
14  |   |   |   every round where node  $i$  is still UNDECIDED;
15  |   | else if  $j$  has output 0 then
16  |   |   | Remove  $j$  from UndecidedNeighbors;
17  | foreach  $j \in \text{UndecidedNeighbors}$  do
18  |   | if message  $s_j(i)$  is  $\phi$  then
19  |   |   | BeenCheated  $\leftarrow$  true;
20  |   |   | Compute  $rps_{i,j}(s_i(j), s_j(i))$  as defined in Definition 3.1;
21  | if  $rps_{i,j}(s_i(j), s_j(i)) = i$  for all  $j \in \text{UndecidedNeighbors}$  then
22  |   | Output 1 (join MIS);
23  | /* Round 3: Observe Undecided Neighbors' Outputs */
24  | if there exists a node in UndecidedNeighbors has output  $\perp$  then
25  |   | From this point onward, disregard the rest of the algorithm and output  $\perp$ 
26  | else if there exists a node in UndecidedNeighbors has output 1 then
27  |   | if there exists a neighbor  $j$  with output 1 and  $rps_{i,j}(s_i(j), s_j(i)) \neq j$  then
28  |   |   | BeenCheated  $\leftarrow$  true;
29  |   | From this point onward, disregard the rest of the algorithm and output  $\perp$  if
30  |   |   | BeenCheated is true, and output 0 otherwise, in every round where node  $i$  is
31  |   |   | still UNDECIDED;
32  | else
33  |   | Remove all  $j \in \text{UndecidedNeighbors}$  who output 0;
34  |   | if UndecidedNeighbors is empty then
35  |   |   | From this point onward, disregard the rest of the algorithm and Output 1 in
36  |   |   |   every round where node  $i$  is still UNDECIDED;
```

- In Round 2, node i outputs 1 only if no neighbor has yet output 1 or \perp , and i has won the RPS game against all its undecided neighbors. Since all neighbors also follow the algorithm, any neighbor that loses to i in the RPS game will not output 1. Thus, no neighbor of i can simultaneously output 1.
- In Round 3, node i outputs 1 only if all of its neighbors have already output 0. Hence, it is impossible for a neighbor to output 1 in this case.

Therefore, Case 2 cannot occur. Combined with the fact that Case 1 is also impossible, we conclude that by following Algorithm 1, node i 's utility is never less than 0. \square

Theorem 3.5. *The strategy corresponding to Algorithm 1 constitutes a Belief-Independent Sequential Equilibrium.*

Proof. Consider an arbitrary history h with $i \in P(h)$. Let h_i denote the local history of node i , which implies $h \in I_i(h_i) \in \mathcal{I}_i$. We show that the continuation strategy prescribed for i by Algorithm 1 from information set $I_i(h_i)$ is sequentially rational irrespective of the i 's belief over $I_i(h_i)$, i.e., it gives the maximum expected payoff when all other nodes also follow Algorithm 1 from h . It follows that the strategy constitutes a sequential equilibrium for any belief system of i . and it follows that the strategy constitutes a Belief-Independent Sequential Equilibrium. As we mentioned in Section 2, we do not consider internal deviations except for randomness generation. We distinguish deviations as:

- *Undetectable deviations:* for example, choosing RPS moves with nonuniform probabilities. Since every sequence of moves is still possible under the prescribed uniform strategy (albeit with small probability), neighbors can never rule out a deviation with certainty, even after many observations. Similarly, voluntarily outputting 0 or \perp earlier than prescribed may also be indistinguishable, since it could be a valid response to another neighbor having output 1, which remaining neighbors may not observe.
- *Detectable deviations:* e.g., failing to send a required/valid message, sending inconsistent messages, or outputting 1 when not eligible. Such deviations are caught by at least one neighbor, upon which the neighbor sets its **BeenCheated** flag to 1.

Case analysis.

Case A: At h , i has a decided neighbor with output 1 or \perp . Then the maximum attainable payoff for i is 0. By Lemma 3.4, following the algorithm yields a payoff of at least 0. Hence, no profitable deviation exists, so the prescribed action is a best response.

Case B: At h , i do not have a decided neighbor with output 1 or \perp and there exists an undecided neighbor that i has cheated. If the remaining undecided neighbors follow the algorithm, Lemma 3.3 implies the most i can obtain is 0; following the algorithm also yields payoff 0. Thus, no profitable deviation exists.

Case C: At h , i does not have a decided neighbor with output 1 or \perp and i has not cheated any undecided neighbor.

- *Detectable deviation by i :* Some neighbor detects it and sets **BeenCheated**= 1. By Lemma 3.3, the best payoff i can then secure is 0. By Lemma 3.4, sticking to the algorithm yields a payoff of at least 0, so no profitable deviation exists.
- *Undetectable deviation by i :* (i) Changing the probabilities with which i plays rock, paper, scissors does not improve i 's win probability against neighbors who pick each move uniformly at random. Therefore, i 's chance of joining the MIS is unchanged, and so is the expected payoff. (ii) Voluntarily outputting 0 or \perp can yield a payoff at most 0, whereas Lemma 3.4 guarantees that following the algorithm yields payoff at least 0. Hence, no strictly profitable deviation exists.

Therefore, we have shown that for every history h with $i \in P(h)$, the prescribed strategy for i based on $h_i = \text{proj}_i(h)$ gives maximum expected utility possible from h when all other nodes are following the algorithm from h , which proves the theorem. \square

We have shown that Algorithm 1 forms a Belief-Independent Sequential Equilibrium. To complete the proof of Theorem 3.2, we now assume that all nodes follow the algorithm without deviation and, under this honest execution, we analyze the simplified version of the algorithm (with deviation-handling logic removed) and proceed to prove its correctness, termination, and nonzero inclusion probability.

3.2 Strategy Algorithm under No Deviations

Since we have established in Theorem 3.5 that Strategy Algorithm 1 constitutes a Belief-Independent Sequential Equilibrium, no node has a unilateral incentive to deviate. When no node deviates from the Strategy Algorithm 1, its execution is equivalent to running Algorithm 2, which is simply Strategy Algorithm 1 with the deviation handling logic removed.

Algorithm 2: RPS based MIS strategy algorithm for node i without deviations

```

1 Initialization
2   Initialize node  $i$  as undecided;
3    $\text{UndecidedNeighbors} \leftarrow N(i)$ ;
4 while true do
5   /* Round 1: Play RPS with Undecided Neighbors */
6   foreach  $j \in \text{UndecidedNeighbors}$  do
7     Choose strategy  $s_i(j) \in \{r, p, s\}$  uniformly at random and send to  $j$ ;
8   /* Round 2: Join MIS if possible */
9   if  $\text{rps}_{i,j}(s_i(j), s_j(i)) = i$  for all  $j \in \text{UndecidedNeighbors}$  then
10    Output 1 (join MIS and terminate);
11  /* Round 3: Observe Undecided Neighbors' Outputs */
12  if any  $j \in \text{UndecidedNeighbors}$  has output 1 then
13    Output 0 and terminate;
14  Remove all  $j \in \text{UndecidedNeighbors}$  who output 0;
15  if  $\text{UndecidedNeighbors}$  is empty then
16    Output 1 (join MIS and terminate);

```

We now proceed to prove the remaining claims of Theorem 3.2 on this deviation-free execution: namely, correctness (that the output forms a valid MIS), termination (with high probability), and positive inclusion probability (every node has a nonzero chance of joining the MIS).

Theorem 3.6 (Termination). *Algorithm 2 terminates in $O(3^{4\Delta} \log n)$ rounds with high probability on graphs with maximum degree Δ .*

Proof. Let Y denote the number of edges removed in a given iteration, and for each edge $\{i, j\} \in E$, define X_{ij} to be an indicator variable for the event that i wins with all its neighbors and j wins with all its neighbors except for i . Let d_i and d_j be the number of undecided neighbors of i and j respectively. As i needs to win a total of d_i games and j needs to win a total of $d_j - 1$ games, we have, $\Pr[X_{ij} = 1] = 1/(3^{d_i+d_j-1})$

We now relate this to the total number of edges that have been removed. Let Y be the total number of removed edges, and let both Y_{ij} and Y_{ji} denote the removal of edge $\{i, j\}$. For all vertices $i \in N(j)$ there can only be one node k with $X_{kj} = 1$, and if there is such a node k , then $Y_{ij} = 1$ for all $i \in N(j)$ (as k will join the MIS and cause j to be deleted and hence all edges

incident on j will be deleted). Therefore, $\sum_{i \in N(j)} d_j X_{ij} \leq \sum_{i \in N(j)} Y_{ij}$. Now summing over all the vertices gives

$$\sum_{j \in V} \sum_{i \in N(j)} d_j X_{ij} \leq \sum_{j \in V} \sum_{i \in N(j)} Y_{ij} \implies \sum_{ij \in E} (d_j X_{ij} + d_i X_{ji}) \leq \sum_{ij \in E} 2Y_{ij}$$

Taking the expectation on both sides,

$$2\mathbb{E}[Y] \geq \sum_{ij \in E} (d_j \cdot \mathbb{E}[X_{ij}] + d_i \cdot \mathbb{E}[X_{ji}]) \geq \sum_{ij \in E} \left(\frac{3d_j}{3^{d_i+d_j}} + \frac{3d_i}{3^{d_j+d_i}} \right) = \sum_{ij \in E} \frac{3(d_i + d_j)}{3^{d_i+d_j}}$$

Let Δ be the maximum degree in the graph. Since $d_i, d_j \leq \Delta$ and $d_i + d_j \geq 2$, we get

$$\mathbb{E}[Y] \geq \frac{3}{2} \sum_{ij \in E} \frac{d_i + d_j}{3^{2\Delta}} = \frac{3}{2 \cdot 3^{2\Delta}} \sum_{ij \in E} (d_i + d_j) \geq \frac{3}{2 \cdot 3^{2\Delta}} \cdot \sum_{ij \in E} 2 = \frac{3|E|}{3^{2\Delta}}$$

Let $\alpha = 3/(2 \cdot 3^{2\Delta})$. Therefore $\mathbb{E}[Y] \geq 2\alpha|E|$. Using Lemma A.1, with $d = \alpha|E|$ and $a = |E|$:

$$\Pr(Y \geq \alpha|E|) \geq \frac{\mathbb{E}[Y] - \alpha|E|}{|E| - \alpha|E|} \geq \frac{\alpha}{1 - \alpha} \geq \alpha$$

Thus, with probability at least $\alpha = 3/(2 \cdot 3^{2\Delta})$, at least an α -fraction of the edges are removed in a single iteration. Let E' be the set of edges remaining after the iteration, we have $|E'| \leq (1 - \alpha)|E|$ with probability at least α .

To reduce the number of edges from $|E| \leq n^2$ to 0, it suffices to have r successful iterations of edges reducing by factor of $(1 - \alpha)$ such that $n^2 \cdot (1 - \alpha)^r \leq n^2 \cdot e^{-\alpha r} < 1$ which implies $r > (2/\alpha) \ln n$. Now we will calculate the number of trials T required to get r successful iterations with high probability.

For each iteration $1 \leq t \leq T$, let Z_t be an indicator variable denoting the event that at least a $(1 - \alpha)$ -fraction of edges were removed in iteration t . Let $Z = \sum_{t=1}^T Z_t$ be the total number of successful rounds out of T total rounds. By linearity of expectations, we have $\mathbb{E}[Z] \geq \alpha T$. Now consider $T = (4c/\alpha^2) \log n$ for some constant $c > 0$. We have $\mathbb{E}[Z] \geq (4c/\alpha) \log n$. Applying the Chernoff bound A.2 with $\delta = 1/2$, we obtain

$$\Pr(Z \leq (2c/\alpha) \log n) \leq \exp\left(-\frac{4c \log n}{8\alpha}\right) \ll n^{-c}$$

So, if we have T trials, we get that whp, more than $(2c/\alpha) \log n$ successful iterations, which implies that all edges are removed and the algorithm terminates. Therefore, the algorithm terminates in $O(3^{4\Delta} \log n)$ rounds with high probability, which proves the lemma. \square

Theorem 3.7 (Correctness). *When all nodes follow Algorithm 2, after termination, the set of nodes that output 1 forms a valid Maximal Independent Set (MIS) of the network graph, and all remaining nodes output 0.*

Proof. We now prove that the output of Algorithm 2 satisfies both the independence and maximality conditions of a Maximal Independent Set (MIS).

- *Independence:* A node i outputs 1 (i.e., joins the MIS) only if it wins the Rock-Paper-Scissors (RPS) game against all its currently undecided neighbors, given that all its decided neighbors have output only 0. In no case, two neighboring nodes i and j both win against each other in the same iteration. Therefore, no two adjacent nodes can simultaneously output 1, and once a node outputs 1, its neighbors are required to output 0 in the next round, thereby ensuring independence.

- *Maximality:* A node i outputs 0 only in Round 3 and only if at least one of its neighbors has output 1. Moreover, each node continues participating until it outputs either 0, 1, or \perp . However, under the assumption that all nodes follow Algorithm 2 honestly, no node ever outputs \perp (as no node deviates). Thus, since termination is guaranteed with high probability by Theorem 3.6, every node eventually outputs 1 or 0, and every node that outputs 0 must be adjacent to a node in the MIS. Hence, the resulting set is maximal. \square

Theorem 3.8 (Positive Probability of MIS Inclusion). *Under Algorithm 2, each node has a non-zero probability of eventually outputting 1, i.e., joining the MIS, in every round where none of its neighbors has output 1*

Proof. In any iteration where a node i is still undecided and none of its neighbors has output 1, it plays the RPS game with all of its undecided neighbors. Since the outcome of RPS is uniformly random, the probability that i wins against any particular neighbor is $\frac{1}{3}$. Consequently, the probability that i wins against all of its undecided neighbors and joins the MIS in that iteration is $(\frac{1}{3})^{d_i}$, where d_i denotes the number of undecided neighbors of i . Thus, as long as none of its neighbors has output 1, node i always has a strictly positive probability of joining the MIS. \square

4 A Faster Strategy Algorithm using Cryptography

The RPS-based algorithm described in the previous section is not very efficient on graphs with high-degree nodes. The exponential dependence on degree arises because each node plays independent RPS games with its neighbors and can only join the MIS if it wins *all* of them. To improve the round complexity, we require an algorithm where each node plays a single game with all of its neighbors, under the condition that if a node wins, none of its neighbors can win. The algorithm by Métivier et al. [15], can be viewed as such a game: each node generates a random rank, and a node “wins” if its rank is lower than that of all its neighbors, in which case it joins the MIS. However, as discussed in Section 1, this algorithm by itself fails in rational settings because nodes can bias their rank selection to gain an advantage.

A natural attempt to get around this obstacle is to have each node’s rank also depend on inputs from its neighbors. For instance, if rank of node i is computed using randomness from both i and an arbitrarily chosen neighbor j , then neither i nor j can fully influence the outcome: i prefers a low rank in order to join the MIS, while j prefers i to have a high rank in order to block it from joining the MIS. This idea of *two-party shared randomness* ensures that the ranks will be *truly random* and cannot be changed unilaterally by any single party. To make this verifiable, nodes must prove the correctness of their computed rank to others. This requires sharing the opponent’s input, which must be authenticated by some means. We perform authentication using digital signatures, which requires standard cryptographic assumptions.

Additional Assumptions As mentioned, our approach relies on cryptographic tools; we assume that each node i is equipped with a public–private key pair and can sign messages. In order to verify the digital signatures, all nodes are assumed to know the identifiers and public keys of every other node in the network. Further, nodes can only broadcast to their neighbors.³

³The algorithm can, in principle, be extended to support unicast communication. However, doing so introduces several corner cases that would require substantially more discussion. One such case arises when a node can avoid detection by selecting multiple opponents simultaneously. Although this strategy does not yield a higher expected payoff than choosing a single opponent, we must still account for it, since the algorithm is required to prescribe the best continuation strategy for every possible history.

4.1 Rationality Resilient Rank-based Strategy Algorithm

The algorithm proceeds in *iterations*, where each iteration consists of *five rounds*. Initially, all nodes are in the **undecided** state. Once a node outputs a value in $\{0, 1, \perp\}$, it becomes **decided**, meaning that its output is visible to all neighbors and it takes no further action in subsequent rounds. Each node maintains a **BeenCheated** flag, initialized to **false**, and in every iteration, the algorithm proceeds through the following rounds:

- **Round 1: Opponent Selection.** Each node i selects an arbitrary undecided neighbor as its *opponent*, denoted by $\text{opp}(i)$, and broadcasts this choice to all neighbors.
- **Round 2 (Randomness Generation and Exchange).** Each node i generates a $c \log n$ length bit string $r_{i \rightarrow i} \in \{0, 1\}^{c \log n}$ uniformly at random and broadcasts it to all neighbors, where $c > 0$ is a fixed constant. For each neighbor j who selected i as an opponent in Round 1, node i generates a separate $c \log n$ length bit string $r_{i \rightarrow j} \in \{0, 1\}^{c \log n}$ uniformly at random, signs it along with the current iteration counter, and broadcasts each signed message.
- **Round 3: Share Randomness of Opponent** Each node i that broadcast $\text{opp}(i)$ in Round 1 and received $r_{\text{opp}(i) \rightarrow i}$ computes its rank as $R_i = r_{i \rightarrow i} \oplus r_{\text{opp}(i) \rightarrow i}$, and then broadcasts the signed message it received from $\text{opp}(i)$ to all its neighbors.
- **Round 4: Calculate Ranks of Neighbors.** Each node i , for each of its neighbors j , after receiving a valid signed $r_{\text{opp}(j) \rightarrow j}$ in the previous round, computes the rank $R_j = r_{j \rightarrow j} \oplus r_{\text{opp}(j) \rightarrow j}$. If any undecided neighbor j did not broadcast some message in the previous 3 rounds as expected, then i assumes j 's rank to be $\{1\}^{c \log n}$ and sets its **BeenCheated** flag to **true**, since such nodes are considered to have deviated from the algorithm. Similarly, if i deviated in a manner that can be detected by some of its neighbors (this includes the case where i fails to broadcast $r_{\text{opp}(i) \rightarrow i}$ because it did not receive it from $\text{opp}(i)$), then it sets its own rank to $\{1\}^{c \log n}$ in order to stay consistent with neighbors that detect i 's deviation, and avoid a potential $-\infty$ payoff.

Next, node i checks whether any neighbor has already output 1 or \perp in the current iteration; if so, it outputs \perp in every subsequent round in which it remains **Undecided**. Otherwise, it compares its rank with all its undecided neighbors. If $R_i < R_j$ for all such neighbors j , then node i outputs 1, joining the MIS and terminates.

- **Round 5: Reacting to Neighbor Decisions.** Each node i checks whether any undecided neighbor j has output \perp . If such a node exists, i outputs \perp in every subsequent round in which it remains **Undecided**.

Else, if any neighbor j has output 1 in previous round, then:

- If, for all such j with output 1, it holds that $R_j < R_i$, then node i outputs 0 in every subsequent round while it remains undecided, unless **BeenCheated** is **true**, in which case it outputs \perp in every subsequent round while it remains undecided..
- Otherwise, if there exists a node j with $R_j > R_i$ that still outputs 1, then i outputs \perp in every subsequent round while it remains undecided.

If there are no **Undecided** neighbors, i outputs 1. otherwise, i proceeds to next iteration.

The distributed strategy algorithm described above is formalized in Algorithm 3. Since the description is quite lengthy, we have modularized it into several algorithms, each describing one round of the iteration, presented in Algorithms 4 – 8. We then proceed to prove the main theorem, Theorem 4.1, which demonstrates that the proposed algorithm is rationality resilient.

Theorem 4.1. *Algorithm 3 is a Rationality Resilient Algorithm as defined in Definition 2.10.*

Algorithm 3: Rank based MIS algorithm for Node i

```
1 Initialization
2    $\text{UndecidedNeighbors} \leftarrow N(i)$ ;  $\text{BeenCheated} \leftarrow \text{false}$ ;  $\text{iteration\_number} \leftarrow 0$ 
3 if  $N(i) = \emptyset$  then
4    $\text{In every round where node } i \text{ is still UNDECIDED, Output 1}$ 
5 while  $i$  is UNDECIDED do
6    $\text{iteration\_number} \leftarrow \text{iteration\_number} + 1$ ;
7   // --- Round 1 ---
8   Run Algorithm Opponent Selection (Algorithm 4) ;
9   // --- Round 2 ---
10  Run Algorithm Generate Randomness (Algorithm 5);
11  // --- Round 3 ---
12  Run Algorithm Forward Opponent's random input (Algorithm 6);
13  // --- Round 4 ---
14  Run Algorithm Calculate Ranks of Neighbors (Algorithm 7);
15  // --- Round 5 ---
16  Run Algorithm Reacting to Neighbor Decisions (Algorithm 8);
```

Algorithm 4: Opponent Selection

```
1 Choose any  $k \in \text{UndecidedNeighbors}$  as the opponent  $\text{opp}(i)$ , and broadcast this
   choice to all neighbors.
```

Algorithm 5: Generate Randomness

```
1 if  $i$  did not broadcast a valid ID in Round 1 then
2    $R_i \leftarrow \{1\}^{c \log n}$ ;
3   // Node  $i$  deviated; as neighbors assume its rank is  $\{1\}^{c \log n}$ ,  $i$  also
4   adopts this to stay consistent and avoid  $-\infty$  payoff.
5   Draw  $c \log n$  bits  $r_{i \rightarrow i} \in \{0, 1\}^{c \log n}$  uniformly at random and broadcast to all the
   neighbors;
6   foreach  $j \in \text{UndecidedNeighbors}$  which broadcasted  $\text{opp}(j) = i$  in previous round
   do
7     Draw a random  $c \log n$  bit  $r_{i \rightarrow j} \in \{0, 1\}^{c \log n}$  uniformly at random and broadcast
     signed message  $(\text{iteration\_number}, i, j, r_{i \rightarrow j})$  to all neighbors;
8   foreach  $j \in \text{UndecidedNeighbors}$  that did not broadcast an opponent in Round 1 do
9     Node  $i$  sets  $R_j \leftarrow \{1\}^{c \log n}$  and  $\text{BeenCheated} \leftarrow \text{true}$ ; /* as  $j$  deviated */
```

Algorithm 6: Forward Opponent's random input

```
1 if  $i$  broadcasted ID of  $\text{opp}(i)$  as its opponent in Round 1 then
2   if valid signed  $r_{\text{opp}(i) \rightarrow i}$  is recieved from  $\text{opp}(i)$  in Round 2 then
3      $R_i \leftarrow \{r_{i \rightarrow i} \oplus r_{\text{opp}(i) \rightarrow i}\}$ ;
4     Broadcast message  $(\text{iteration\_number}, \text{opp}(i), r_{\text{opp}(i) \rightarrow i})$  signed by  $\text{opp}(i)$  to all
     neighbors
5   else
6      $\text{BeenCheated} \leftarrow \text{true}$ ; /*  $i$  was cheated by  $\text{opp}(i)$  */
7      $R_i \leftarrow \{1\}^{c \log n}$ ; /* as  $i$  does not have signed message from  $\text{opp}(i)$ , its
     neighbors will assign this rank to  $i$  */
```

Algorithm 7: Calculate Ranks of Neighbors

```
1 if any  $j \in \text{UndecidedNeighbors}$  has output 1 or  $\perp$  then
2   | From this point onward, disregard the rest of the algorithm and output  $\perp$  in every
   | round where node  $i$  is still UNDECIDED;
3 else
4   | Remove all  $j \in \text{UndecidedNeighbors}$  that output 0 from UndecidedNeighbors;
5 foreach  $j \in \text{UndecidedNeighbors}$  do
6   | if message (iteration_number,  $\text{opp}(j)$ ,  $r_{\text{opp}(j) \rightarrow j}$ ) signed by  $\text{opp}(j)$  is received from  $j$ 
   | then
7   |   | Node  $i$  sets  $R_j \leftarrow \{r_{\text{opp}(j) \rightarrow j} \oplus r_{j \rightarrow j}\}$ ;
8   | else
9   |   | Node  $i$  sets  $R_j \leftarrow \{1\}^{c \log n}$ ,  $\text{BeenCheated} \leftarrow \text{true}$ ;          /* as  $j$  deviated */
10 if  $R_i < R_j$  for all  $j \in \text{UndecidedNeighbors}$  then
11   | Output 1 (join MIS);
```

Algorithm 8: Reacting to Neighbor Decisions

```
1 if any node in UndecidedNeighbors has output 1 then
2   | if for all such  $j$  with output 1,  $R_j < R_i$  then
3   |   | From this point onward, disregard the rest of the algorithm; if  $\text{BeenCheated}$  is
   |   | true, then output  $\perp$ , else output 0, in every round where node  $i$  is still
   |   | UNDECIDED.
4   | else
5   |   | From this point onward, disregard the rest of the algorithm and Output  $\perp$  in
   |   | every round where node  $i$  is still UNDECIDED
6 else
7   | Remove all  $j \in \text{UndecidedNeighbors}$  who output 0
8   | if UndecidedNeighbors is empty then
9   |   | From this point onward, disregard the rest of the algorithm and Output 1 in
   |   | every round where node  $i$  is still UNDECIDED
```

We prove Theorem 4.1 in three parts (Theorems 4.4, 4.5, and 4.6). Among these, Theorem 4.4 requires intermediate results, which we establish through Lemmas 4.2 and 4.3.

Lemma 4.2. *Let the game be at an arbitrary history h with $i \in P(h)$, where $h \in I_i \in \mathcal{I}_i$. Suppose that one of i 's **UndecidedNeighbors** (say, j) has its **BeenCheated** flag set to **true**. Then, regardless of the history h and of i 's strategy from h onward, if all the other nodes follow Algorithm 3 from h onward, the utility of node i will be at most 0.*

Proof. According to Algorithm 3, node j outputs 0 only in Round 5 of an iteration, and even then, only if its **BeenCheated** flag is **false**. Since the **BeenCheated** flag, once set to **true**, is never reset to **false**, a node with **BeenCheated** = **true** will never output 0. Therefore, node j will only output 1 or \perp . In either case, the utility of i is 0. \square

Lemma 4.3. *Let the game be at an arbitrary history h with $i \in P(h)$, where $h \in I_i \in \mathcal{I}_i$, and let h_i be the local history of node i . Then, provided that all the undecided nodes follow Algorithm 3 from this point onward, the utility of node i is at least 0.*

Proof. A node i receives a utility less than zero (i.e., $-\infty$) only in two cases:

1. i outputs 0, and none of its neighbors also outputs 1, or
2. i outputs 1, and at least one of its neighbors also outputs 1.

Case 1 is not possible under the algorithm: a node is instructed to output 0 only in Round 4, and only after observing at least one neighbor output 1. Hence, with any history where the node i is undecided in Round 3 outputs 0, it must be the case that a neighbor has already output 1. Therefore, the first case cannot occur.

Case 2 is also prevented by the algorithm: a node outputs 1 only in Round 3 and only if:

- No neighbor has output 1 or \perp in the previous rounds, and
- i has the least rank compared to all its undecided Neighbors

Because all Undecided neighbors follow the algorithm, the nodes with rank less than i will not output 1 – they will either output 0 or \perp in later Rounds.

Therefore, following the algorithm ensures that node i 's utility is never less than 0. \square

Theorem 4.4. *Algorithm 3 constitutes a Belief-Independent Sequential Equilibrium.*

Proof. Consider an arbitrary history h with $i \in P(h)$. Let h_i denote the local history of node i , which implies $h \in I_i(h_i) \in \mathcal{I}_i$. We show that the continuation strategy prescribed for i by Algorithm 3 from information set $I_i(h_i)$ is sequentially rational irrespective of the i 's belief over $I_i(h_i)$, i.e., it gives the maximum expected payoff when all other nodes also follow Algorithm 3 from h . It follows that the strategy constitutes a sequential equilibrium for any belief system of i . and it follows that the strategy constitutes a Belief-Independent Sequential Equilibrium. As we mentioned in Section 2, we do not consider internal deviations; this indirectly means that node i knows exactly which line/round of the algorithm applies at h_i . We distinguish deviations as

- *Undetectable deviations:* for example, choosing internal randomness $r_{i \rightarrow i}, r_{i \rightarrow j}$ with nonuniform probabilities. Since every sequence of ranks is still possible under the prescribed uniform strategy (albeit with small probability), neighbors can never rule out a deviation with certainty, even after many observations. Similarly, voluntarily outputting 0 or \perp earlier than prescribed may also be indistinguishable, since it could be a valid response to another neighbor having output 1, which remaining neighbors may not observe.

- *Detectable deviations:* e.g., failing to send a required/valid message, sending inconsistent messages, or outputting 1 when not eligible. Such deviations are caught by at least one neighbor, which sets its **BeenCheated** flag to 1.

Case analysis.

Case A: At h , i has a decided neighbor with output 1 or \perp . Then the maximum attainable payoff for i is 0. By Lemma 4.3, following the algorithm yields a payoff of at least 0. Hence, no profitable deviation exists, so the prescribed action is a best response.

Case B: At h , i do not have a decided neighbor with output 1 or \perp and there exists an undecided neighbor that i has cheated. If the remaining undecided neighbors follow the algorithm, Lemma 4.2 implies the most i can obtain is 0; following the algorithm also yields payoff 0. Thus, no profitable deviation exists.

Case C: At h , i do not have a decided neighbor with output 1 or \perp and i has not cheated any undecided neighbor.

- *Detectable deviation by i :* Some neighbor detects it and sets **BeenCheated** = 1. By Lemma 4.2, the best payoff i can then secure is 0. By Lemma 4.3, sticking to the algorithm yields payoff at most 0 and at least 0, so no profitable deviation exists.
- *Undetectable deviation by i :* (1) Changing RPS mixing probabilities does not improve i 's win probability against neighbors who randomize uniformly; i 's chance of joining the MIS is unchanged, so expected payoff is unchanged. (2) Voluntarily outputting 0 or \perp can yield at most 0, whereas Lemma 4.3 guarantees sticking yields payoff at least 0. Hence, no profitable deviation exists.

Therefore, we have shown that for every history h with $i \in P(h)$, the prescribed strategy for i based on $h_i = \text{proj}_i(h)$ gives maximum expected utility possible from h when all other nodes are following the algorithm from h , which proves the theorem. \square

We have shown that Algorithm 3 forms a Belief-Independent Sequential Equilibrium. To complete the proof of Theorem 3.2, we now assume that all nodes follow the algorithm without deviation. Under this honest execution, we analyze the simplified version of the algorithm (with deviation-handling logic removed) and proceed to prove its correctness, termination, and nonzero inclusion probability.

4.2 Strategy Algorithm without Deviations

Since we have established in Theorem 4.4 that Strategy Algorithm 3 constitutes a Belief-Independent Sequential Equilibrium, no node has a unilateral incentive to deviate. When no node deviates from the Strategy Algorithm 3, its execution is equivalent to running Algorithm 9, which is simply Strategy Algorithm 3 with the deviation handling logic removed.

We now proceed to prove the remaining claims of Theorem 4.1 on this deviation-free execution: namely, correctness (that the output forms a valid MIS), termination (with high probability), and positive inclusion probability (every node has a nonzero chance of joining the MIS).

Theorem 4.5 (Correctness and Termination). *Algorithm 9 terminates in $O(\log n)$ rounds with high probability, and the set of nodes that output 1 at termination forms a valid Maximal Independent Set (MIS).*

Proof. Algorithm 9 simulates the algorithm by Métivier et al. [15] through verifiable random priority generation and decision-making rounds.

Each iteration of Algorithm 9 consists of five communication rounds. This five-round process mirrors one round of the algorithm by Métivier et al. [15]. Since that algorithm terminates in $O(\log n)$ rounds with high probability and outputs a correct MIS, our algorithm also inherits

Algorithm 9: Rank based MIS strategy algorithm for node i under no deviations

```

1 Initialization
2    $\lfloor$  UndecidedNeighbors  $\leftarrow N(i)$ ; iteration_number  $\leftarrow 0$ 
3 if  $N(i) = \emptyset$  then
4    $\lfloor$  Output 1
5 while  $i$  is UNDECIDED do
6   iteration_number  $\leftarrow$  iteration_number + 1;
7   /* Round 1: Opponent Selection */
8   Choose any  $k \in$  UndecidedNeighbors as the opponent opp( $i$ ), and broadcast this
   choice to all neighbors.
9   /* Round 2: Priority Computation */
10  Draw random  $c \log n$  bit  $r_{i \rightarrow i} \in \{0, 1\}^{c \log n}$  uniformly at random and broadcast to all
   the neighbors;
11  foreach  $j \in$  UndecidedNeighbors which broadcasted opp( $j$ ) =  $i$  in previous round
   do
12    Draw a random  $c \log n$  bit  $r_{i \rightarrow j} \in \{0, 1\}^{c \log n}$  uniformly at random and broadcast
    signed message (iteration_number,  $i$ ,  $r_{i \rightarrow j}$ ) to all neighbors;
13  /* Round 3: Forward Opponent's random input */
14   $R_i \leftarrow \{r_{i \rightarrow i} \oplus r_{\text{opp}(i) \rightarrow i}\}$ 
   Broadcast message (iteration_number, opp( $i$ ),  $r_{\text{opp}(i) \rightarrow i}$ ) signed by opp( $i$ ) to all
   neighbors
15  /* Round 4: Calculate Ranks of Neighbors */
16  foreach  $j \in$  UndecidedNeighbors do
17     $\lfloor$   $R_j \leftarrow \{r_{\text{opp}(j) \rightarrow j} \oplus r_{j \rightarrow j}\}$ ;
18  if  $R_i < R_j$  for all  $j \in$  UndecidedNeighbors then
19     $\lfloor$  Output 1 (join MIS);
20  /* Round 5: Reacting to Neighbor Decisions */
21  if any  $j \in$  UndecidedNeighbors has output 1 then
22     $\lfloor$  Output 0
23  else
24    Remove all  $j \in$  UndecidedNeighbors who output 0;
25    if UndecidedNeighbors is empty then
26       $\lfloor$  Output 1

```

these guarantees. Thus, Algorithm 9 terminates in $5 \times O(\log n)$ iterations (i.e., $O(\log n)$ rounds asymptotically). However, an additional argument is needed: in the original algorithm by Métivier et al. [15], the probability that two nodes pick the same rank is negligible due to the use of real-valued randomness. In contrast, in our algorithm, each node generates a rank using $c \log n$ random bits, so the probability that any two nodes pick the same rank is non-negligible.

In each round, every node generates a rank consisting of $c \log n$ random bits. The total number of possible ranks is $2^{c \log n} = n^c$. By the union bound, the probability that any two nodes pick the same rank in a given round is at most $\binom{n}{2} \cdot \frac{1}{n^c} = O\left(\frac{1}{n^{c-2}}\right)$. When this process is repeated independently for $O(\log n)$ rounds, the probability that any two nodes collide in at least one of the rounds is bounded by $O\left(\frac{\log n}{n^{c-2}}\right)$, which is negligible for any sufficiently large constant $c > 2$. Thus, with high probability, no two nodes ever pick the same rank across all rounds.

Since the original algorithm with real-valued ranks succeeds with probability $1 - 1/\text{poly}(n)$, the bit-based version succeeds with probability at least $1 - 1/\text{poly}(n)$. \square

Theorem 4.6 (Positive Probability of MIS Inclusion). *Under Algorithm 9, every node has a strictly positive probability of eventually outputting 1 (i.e., joining the MIS) in each round in which none of its neighbors has output 1.*

Proof. Consider an undecided node i . Its priority is computed as $R_i = r_{i \rightarrow i} \oplus r_{\text{opp}(i) \rightarrow i}$, where both random values are drawn independently and uniformly from $\{0, 1\}^{c \log n}$. The priorities of all neighbors are defined analogously, so that they are independent and identically distributed over $\{0, 1\}^{c \log n}$.

Hence, the probability that R_i is strictly smaller than the priorities of all its active neighbors is $1/d_i$ where d_i is the number of undecided neighbors of i . Therefore, as long as none of its neighbors has the output 1, node i always has a strictly positive probability of joining the MIS. \square

5 Conclusion and Future Work

We proposed a model that incorporates rational behavior in distributed message passing algorithms. We considered a utility function, where nodes are incentivized to compute a correct solution and prefer to be included in the MIS. We designed two algorithms that are resilient to rational behavior: no node has an incentive to deviate from the prescribed algorithm. Additionally, assuming that no node deviates from the algorithm, we can guarantee correctness, termination, and positive probability of inclusion.

However, each algorithm comes with certain drawbacks. The rank-based strategy algorithm relies on cryptographic assumptions, which implicitly restrict agents to bounded computational power – a non-standard assumption in the context of distributed algorithms. Moreover, this algorithm may use large messages; a node that becomes the opponent of k neighbors will send $O(k \log n)$ -bit messages over each incident edge. On the other hand, the RPS-based strategy algorithm avoids cryptographic assumptions and sends at most $O(\log n)$ -bit messages over each edge per round. But its round complexity has exponential dependence on the maximum degree of the network.

While our algorithms are resilient to unilateral deviations, they do not prevent coordination or implicit collusion among multiple nodes. For example, once a node realizes that it cannot join the MIS (because one of its neighbors has already output 1), it may still have an incentive to continue participating in the algorithm in a way that strategically reduces the chances of its neighbors, thereby improving the inclusion probability of the nodes with which it is colluding. Additionally, the two algorithms presented are somewhat sensitive to the assumptions about

the underlying model and utility structure. If the utilities are perturbed even slightly, our equilibrium guarantees may not hold. For instance, if the utility of a node in a locally valid solution where it belong to the MIS is negative, our algorithms no longer ensure equilibrium. An important direction for future work, therefore, is to design algorithms that remain robust under a broader class of utility functions, collusion, or to establish *impossibility results* that characterize the precise limitations of such robustness.

References

- [1] Ittai Abraham, Danny Dolev, Rica Gonen, and Joseph Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In Eric Ruppert and Dahlia Malkhi, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, pages 53–62. ACM, 2006. doi:10.1145/1146381.1146393.
- [2] Ittai Abraham, Danny Dolev, and Joseph Y. Halpern. Distributed protocols for leader election: A game-theoretic perspective. *ACM Trans. Economics and Comput.*, 7(1):4:1–4:26, 2019. doi:10.1145/3303712.
- [3] Yackolley Amoussou-Guenou, Bruno Biais, Maria Potop-Butucaru, and Sara Tucci Piergiovanni. Rational vs byzantine players in consensus-based blockchains. In Amal El Fallah Seghrouchni, Gita Sukthankar, Bo An, and Neil Yorke-Smith, editors, *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, pages 43–51. International Foundation for Autonomous Agents and Multiagent Systems, 2020. URL: <https://dl.acm.org/doi/10.5555/3398761.3398772>.
- [4] Simon Collet, Pierre Fraigniaud, and Paolo Penna. Equilibria of Games in Networks for Local Tasks. In *22nd International Conference on Principles of Distributed Systems (OPODIS 2018)*, volume 125, pages 6:1–6:16, Dagstuhl, Germany, 2019. URL: <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.OPODIS.2018.6>, doi:10.4230/LIPIcs.OPODIS.2018.6.
- [5] Joan Feigenbaum and Scott Shenker. Distributed algorithmic mechanism design: recent results and future directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M 2002)*, Atlanta, Georgia, USA, September 28-28, 2002, pages 1–13. ACM, 2002. doi:10.1145/570810.570812.
- [6] Jeremy T. Fineman, Calvin C. Newport, Micah Sherr, and Tonghe Wang. Fair maximal independent sets. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, May 19-23, 2014*, pages 712–721. IEEE Computer Society, 2014. doi:10.1109/IPDPS.2014.79.
- [7] Drew Fudenberg and Jean Tirole. *Game theory (3. pr.)*. MIT Press, 1991.
- [8] Mohsen Ghaffari and Christoph Grunau. Near-Optimal Deterministic Network Decomposition and Ruling Set, and Improved MIS . In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2148–2179, Los Alamitos, CA, USA, October 2024. IEEE Computer Society. URL: <https://doi.ieeecomputersociety.org/10.1109/FOCS61266.2024.00007>, doi:10.1109/FOCS61266.2024.00007.
- [9] Tiantian Gong and Zeyu Liu. Rational secret sharing with competition. *IACR Cryptol. ePrint Arch.*, page 242, 2025. URL: <https://eprint.iacr.org/2025/242>.

- [10] Joseph Y. Halpern and Vanessa Teague. Rational secret sharing and multiparty computation: Extended abstract. *CoRR*, abs/cs/0609035, 2006. URL: <http://arxiv.org/abs/cs/0609035>, arXiv:cs/0609035.
- [11] Juho Hirvonen and Sara Ranjbaran. Designing local distributed mechanisms. *CoRR*, abs/2411.06788, 2024. URL: <https://doi.org/10.48550/arXiv.2411.06788>, arXiv:2411.06788, doi:10.48550/ARXIV.2411.06788.
- [12] Nathan Linial. Distributive graph algorithms-global solutions from local data. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 331–335. IEEE Computer Society, 1987. doi:10.1109/SFCS.1987.20.
- [13] Michael Luby. A simple parallel algorithm for the maximal independent set problem. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 1–10. ACM, 1985. doi:10.1145/22145.22146.
- [14] Shaik Maleka, Amjed Shareef, and C. Pandu Rangan. The deterministic protocol for rational secret sharing. In *22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, Miami, Florida USA, April 14-18, 2008*, pages 1–7. IEEE, 2008. doi:10.1109/IPDPS.2008.4536558.
- [15] Yves Métivier, John Michael Robson, Nasser Saheb-Djahromi, and Akka Zemmari. An optimal bit complexity randomized distributed MIS algorithm. *Distributed Comput.*, 23(5-6):331–340, 2011. URL: <https://doi.org/10.1007/s00446-010-0121-5>, doi:10.1007/S00446-010-0121-5.
- [16] Noam Nisan. Algorithms for selfish agents. In Christoph Meinel and Sophie Tison, editors, *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, volume 1563 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1999. doi:10.1007/3-540-49116-3_1.
- [17] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, Cambridge, MA, 1994. URL: <https://sites.math.rutgers.edu/~zeilberg/EM20/OsborneRubinsteinMasterpiece.pdf>.
- [18] Li-Hsing Yen, Jean-Yao Huang, and Volker Turau. Designing self-stabilizing systems using game theory. *ACM Trans. Auton. Adapt. Syst.*, 11(3):18:1–18:27, 2016. doi:10.1145/2957760.
- [19] Li-Hsing Yen and Guang-Hong Sun. Game-theoretic approach to self-stabilizing minimal independent dominating sets. In Yang Xiang, Jingtao Sun, Giancarlo Fortino, Antonio Guerrieri, and Jason J. Jung, editors, *Internet and Distributed Computing Systems - 11th International Conference, IDCS 2018, Tokyo, Japan, October 11-13, 2018, Proceedings*, volume 11226 of *Lecture Notes in Computer Science*, pages 173–184. Springer, 2018. doi:10.1007/978-3-030-02738-4_15.

A Useful Tail Inequalities

Following are two tail inequalities we use in our analysis of the running time of the RPS based MIS algorithm.

Lemma A.1 (Reverse Markov Inequality). *Let Y be a real-valued random variable such that $Y \leq a$ almost surely, for some constant $a \in \mathbb{R}$, and let $d < \mathbb{E}[Y]$. Then,*

$$\Pr(Y \geq d) \geq \frac{\mathbb{E}[Y] - d}{a - d}.$$

Lemma A.2 (Chernoff Bound: Lower Tail). *Let X_1, X_2, \dots, X_n be independent Bernoulli random variables taking values in $\{0, 1\}$, and let $X = \sum_{i=1}^n X_i$, and $\mu = \mathbb{E}[X]$. Then, for any $0 < \delta < 1$, the following inequality holds:*

$$\Pr(X \leq (1 - \delta)\mu) \leq \exp\left(-\frac{\delta^2\mu}{2}\right)$$