The Role of A-priori Information in Networks of Rational Agents

Yehuda Afek
Tel-Aviv University, Tel-Aviv, Israel
afek@post.tau.ac.il

Shaked Rafaeli
Tel-Aviv University, Tel-Aviv, Israel
shakedr@mail.tau.ac.il

Moshe Sulamy
Tel-Aviv University, Tel-Aviv, Israel
moshe.sulamy@cs.tau.ac.il

Abstract

Until now, distributed algorithms for rational agents have assumed a-priori knowledge of n, the size of the network. This assumption is challenged here by proving how much a-priori knowledge is necessary for equilibrium in different distributed computing problems. Duplication – pretending to be more than one agent – is the main tool used by agents to deviate and increase their utility when not enough knowledge about n is given.

We begin by proving that when no information on n is given, equilibrium is impossible for both Coloring and Knowledge Sharing. We then provide new algorithms for both problems when n is a-priori known to all agents. However, what if agents have partial knowledge about n? We provide tight upper and lower bounds that must be a-priori known on n for equilibrium to be possible in Leader Election, Knowledge Sharing, Coloring, Partition and Orientation.

2012 ACM Subject Classification Theory of computation → Distributed computing models

Keywords and phrases rational agents, distributed game theory, coloring, knowledge sharing

Digital Object Identifier 10.4230/LIPIcs.DISC.2018.5

Related Version A full version of the paper is available at [6], `https://arxiv.org/abs/1711.04728`.

Funding This research was supported by the Israel Science Foundation (grant 1386/11).

Acknowledgements We would like to thank Doron Mukhtar for showing us the ring partition problem and proving it is *unbounded*, when we thought such problems do not exist. We would also like to thank Michal Feldman, Amos Fiat, and Yishay Mansour for helpful discussions.

1 Introduction

The complexity and simplicity of most distributed computing problems depend on the inherent a-priori knowledge given to all participants. Usually, the more information processors in a network start with, the more efficient and simple the algorithm for a problem is. Sometimes, this information renders an otherwise unsolvable problem, solvable.

In game-theoretic distributed computing, algorithms run in a network of *rational agents* that may deviate from an algorithm if they deem the deviation more profitable for them. Rational agents have always been assumed to know the number of participants in the network [1, 4, 7, 24, 43], when in fact this assumption is not only very unrealistic in today's Internet, but also provides agents with non-trivial information which is critical for equilibrium.

Consider for example a large world-wide social network on which a distributed algorithm between a large portion of its members is run. It does not necessarily have the time to verify the number of participants, or the service it provides with the algorithm will be irrelevantly slow. If n is known to all participants, as was assumed in previous works about rational agents, that would not be a problem. However, what if n is not known beforehand and this allows one of the participants to skew the result in his favor?

The problems we examine here can be solved in the game-theoretic setting when n is a-priori known. However, learning the size of the network reliably is not possible with rational agents and thus we show that some a-priori knowledge of n is critical for equilibrium. That is, without any knowledge of n, equilibrium for these problems is impossible. In contrast, these problems can be solved without knowledge of n if the participants are not rational, since we can acquire the size of the network using broadcast and echo.

When n is not a-priori known, agents may deviate from the algorithm by *duplicating* themselves to affect the outcome. This deviation is also known as a Sybil Attack [20], commonly used to manipulate internet polls, increase page rankings in Google [15] and affect reputation systems such as eBay [14, 16]. In this paper, we use a Sybil Attack to prove impossibility of equilibria. For each problem presented, an equilibrium when n is known is provided here, or in previous work. Thus when n is *not* known an agent must duplicate to increase its utility, or otherwise if no agent duplicates and the network is 2-connected, a simple broadcast and echo would reveal the actual network size n and the existing equilibrium would apply. Obviously, deviations from the algorithm that include both duplicating and additional cheating, such as lying about the input of duplicated agents or fixing the result of a random coin flip, are also possible. When considering a deviation, an agent assumes it is the only deviating agent, and we assume that there are no coalitions of cheating agents.

Intuitively, the more agents an agent is disguised as, the more power to affect the output of the algorithm it has. For every problem, we strive to find the maximum number of duplications a cheater may be allowed to duplicate without gaining the ability to affect the output, i.e., equilibrium is still possible. This maximum number of duplications depends on whether other agents will detect that a duplication has taken place, since the network could not possibly be this large. To detect this situation they need to possess some knowledge about the network size, or about a specific structure.

We translate this intuition into a precise relation between the lower bound α and the upper bound $\beta \geq \alpha$ on n, that must be a-priori known in order for equilibrium to be possible. We denote this relation f-bound. These bounds hold for both deterministic and non-deterministic algorithms.

These bounds show what algorithms may be used in specific networks. For example, in an internal business network, some algorithms may work because every member in the network knows there are no more than several thousand computers in the network, while for other algorithms this knowledge is not tight enough.

Table 1 summarizes our contributions and related previous work (where there is a citation). Known n refers to algorithms in which n is a-priori known to all agents. Unknown n refers to algorithms or impossibility of equilibrium when agents a-priori know *no* bound on n. The f-bound for each problem is a function f for which there is an equilibrium when the a-priori bounds on n satisfy $\alpha \leq \beta \leq f(\alpha)$, and no equilibrium exists when $\beta > f(\alpha)$. A problem is ∞-bound if there is an equilibrium given *any* finite bound, but no equilibrium exists if no bound or information about n is a-priori given. A problem is *unbounded* if there is an equilibrium even when neither n nor any bound on n is given.

Table 1 Summary of paper contributions, equilibria and impossibility results for different problems with different a-priori knowledge about n.
* f-bound proven for a ring graph, otherwise holds for any 2-connected graph

Problem	Known n	Unknown n	f-bound
Coloring	✓ Section 4	Impossible Section 3	∞*
Leader Election	✓ ADH'13 [4]	Impossible ADH'13 [4]	$(\alpha+1)$
Knowledge Sharing	✓ AGLS'14 [7]	Impossible Section 3	$(2\alpha-2)$*
2-Knowledge Sharing			∞
Partition, Orientation	✓ Section 5	✓ Section 5	Unbounded

1.1 Related Work

The connection between distributed computing and game theory stemmed from the problem of secret sharing [37]. Further works continued the research on secret sharing and multiparty computation when both Byzantine and rational agents are present [2, 18, 21, 22, 23, 31].

Another line of research presented the BAR model (Byzantine, acquiescent and rational) [8, 33, 42], while a related line of research discusses converting solutions with a mediator to cheap talk [2, 3, 12, 13, 19, 27, 32, 38, 40, 41].

Abraham, Dolev, and Halpern [4] were the first to present protocols for networks of rational agents, specifically protocols for Leader Election. In [7] the authors continue this line of research by providing basic building blocks for game theoretic distributed algorithms, namely a wake-up and knowledge sharing equilibrium building blocks. Algorithms for consensus, renaming, and leader election are presented using these building blocks. Consensus was researched further by Halpern and Vilaçça [24], who showed that there is no ex-post Nash equilibrium, and a Nash equilibrium that tolerates f failures under some minimal assumptions on the failure pattern. Yifrach and Mansour [43] studied fair Leader Election protocols, giving an almost tight resilience analysis. Bank, Sulamy, and Waserman [11] examined the case where the *id* space is limited, calculating the minimal threshold for equilibrium.

Coloring and Knowledge Sharing have been studied extensively in a distributed setting [9, 10, 17, 26, 28, 29, 39]. An algorithm for Knowledge Sharing with rational agents was presented in [7], while Coloring with rational agents has not been studied previously, to the best of our knowledge.

Distributed algorithms in which n is not known either implicitly or explicitly have been extensively studied in many other contexts, see for example [5, 25]. In last year's DISC in the permisionless network model and the context of consensus for blockchain [34, 35, 36] similar bounds (factor 2 in their case) on the number of cheating agents have been proved for the consensus task, in the synchronous case.

2 Model

We use the standard message-passing, synchronous model, where the network is a bidirectional graph $G = (V, E)$ with n nodes, each node representing an agent with unlimited computational power, and $|E|$ edges over which they communicate in rounds. G is assumed to be 2-vertexconnected[1]. Throughout the entire paper, n always denotes the *actual* number of nodes in the network.

Initially, each agent knows its own *id* and input, but not the *id* or input of any other agent. For any information that an agent does not know, we assume its prior is uniformly distributed over all possible values. For example, considering the prior of an agent over the *id*s of all other agents, at round 0 each possible permutation of the $n-1$ *ids* in the network is equally possible. Similarly for all possible sets of input vectors, preference vectors, network size, etc. Furthermore, we assume all agents start the protocol together at round 0, i.e., all agents wake-up at the same time. If not, we can use the Wake-Up [7] building block to relax this assumption.

2.1 Equilibrium in Distributed Algorithms

Informally, a distributed algorithm is an equilibrium if no agent at no point in the execution can do better by unilaterally deviating from the algorithm. When considering a deviation, an agent assumes all other agents follow the algorithm, i.e., it assumes it is the only agent deviating. We assume there are no coalitions of cheating agents.

Formally, let o_a be the output of agent a, let Θ be the set of all possible output vectors, and denote the output vector $O = (o_1, \dots, o_n) \in \Theta$, where $O[a] = o_a$. Let Θ_L be the set of *legal* output vectors, in which the protocol terminates successfully, and let Θ_E be the set of *erroneous* output vectors, such that $\Theta = \Theta_L \cup \Theta_E$ and $\Theta_L \cap \Theta_E = \varnothing$.

Each agent a has a utility function $u_a : \Theta \to \mathbb{N}$. The higher the value assigned by u_a to an output vector, the better this vector is for a. As in previous works [4, 7, 43], the utility function is required to satisfy the *Solution Preference* which guarantees that an agent never has an incentive to fail the algorithm. Otherwise, they would simply be Byzantine faults. An agent fails the algorithm only when it detects that another agent had deviated.

▶ **Definition 1** (Solution Preference). The utility function u_a of an agent a never assigns a higher utility to an erroneous output than to a legal one, i.e.:

$$\forall a, O_L \in \Theta_L, O_E \in \Theta_E : u_a(O_L) \geq u_a(O_E)$$

We differentiate the *legal* output vectors, which ensure the output is valid and not erroneous, from the *correct* output vectors, which are output vectors that are a result of a correct execution of the algorithm, i.e., without any deviation. For example, in a Consensus protocol that decides by a majority and a network where the majority of agents received 1 as input and at least one agent received 0, deciding on 0 is *legal*, as it is a valid output for Consensus, but *incorrect*, as it necessarily resulted in a deviation from the protocol in use. The Solution Preference guarantees agents never prefer an erroneous output. However, they may prefer a *legal* but *incorrect* output.

The Solution Preference property introduces the threat agents face when deviating: Agents know that if another agent catches them cheating, it outputs $\perp$ and the algorithm fails. In other words, the output is erroneous, i.e., in Θ_E.

For simplicity, we assume agents only have preferences over their own output, i.e., for any $O_1, O_2 \in \Theta_L$ in which $O_1[a] = O_2[a]$, $u_a(O_1) = u_a(O_2)$. Additionally, each agent a has a

[1] This property was shown necessary in [7], since if a bottleneck node exists it can alter any message passing through it. Such a deviation cannot be detected since all messages between the sub-graphs this node connects must traverse through it. This node can then skew the algorithm according to its preferences.

single preferred output value p_a, and we normalize the utility function values, such that[2]:

$$u_a(O) = \begin{cases} 1 & o_a = p_a \text{ and } O \in \Theta_L \\ 0 & o_a \neq p_a \text{ or } O \in \Theta_E \end{cases} \tag{1}$$

Our results hold for *any* utility function that satisfies Solution Preference. For clarity and ease of presentation we assume Equation 1.

▶ **Definition 2** (Expected Utility). Let r be a round in a specific execution of an algorithm. Let a be an arbitrary agent. For each possible output vector O, let $x_O(s,r)$ be the probability, estimated by agent a at round r, that O is output by the algorithm if a takes step s [3], and all other agents follow the algorithm. The *Expected Utility* a estimates for step s in round r of that specific execution is:

$$\mathbb{E}_{s,r}[u_a] = \sum_{O \in \Theta} x_O(s,r) \cdot u_a(O)$$

An agent will deviate whenever the deviating step has a strictly higher expected utility than the expected utility of the next step of the algorithm, even if that deviating step also increases the risk of an erroneous output.

Let Λ be an algorithm. If by deviating from Λ and taking step s, the expected utility of a is higher, we say that agent a has an *incentive to deviate* (i.e., cheat). For example, at round r algorithm Λ may dictate that a flips a fair binary coin and sends the result to all of its neighbors. Any other action by a is considered a *deviation*: whether the message was not sent to all neighbors, sent later than it should have, or whether the coin toss was not fair, e.g., a only sends 0 instead of a random value. If no agent can unilaterally increase its expected utility by deviating from Λ, we say that the protocol is an *equilibrium*. Equilibrium is defined over a *single* deviating agent, i.e., there are no coalitions of agents.

▶ **Definition 3** (Distributed Equilibrium). Let $s(r)$ denote the next step of algorithm Λ in round r. Λ is an equilibrium if for any deviating step $\bar{s}$, at any round r of every possible execution of Λ:

$$\forall a, r, \bar{s} : \mathbb{E}_{s(r),r}[u_a] \geq \mathbb{E}_{\bar{s},r}[u_a]$$

2.2 Knowledge Sharing

The Knowledge Sharing problem (adapted from [7]) is defined as follows:

1. Each agent a has a private input i_a, in addition to its id, and a function q, where q is identical at all agents.
2. A Knowledge Sharing protocol terminates *legally* if all agents output the *same* value, i.e., $\forall a, b : o_a = o_b \neq \bot$. Thus the set Θ_L is defined as: $O \in \Theta_L \iff \forall a, b : O(a) = O(b) \neq \bot$.
3. A Knowledge Sharing protocol terminates *correctly* (as described in Section 2.1) if each agent outputs at the end the value $q(I)$ over the input values $I = \{i_1, \ldots, i_n\}$ of all other agents[4].

[2] This is the weakest assumption since it still leaves a cheating agent with the highest incentive to deviate, while still satisfying Solution Preference. A utility assigning a lower value for failure than $o_a \neq p_a$ would deter a cheating agent from deviating.

[3] A step specifies the entire operation of the agent in a round. This may include drawing a random number, performing any internal computation, and the contents and timing of any message delivery.

[4] Notice that any output is legal as long as it is the output of all agents, but only a single output value is considered *correct* for a given input vector.

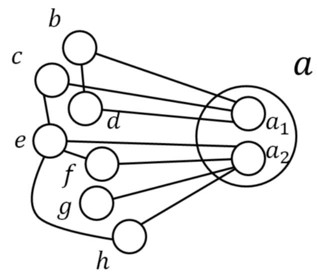

Figure 1 Agent a acting as separate agents a_1, a_2.

4. The function q satisfies the Full Knowledge property:

▶ **Definition 4** (Full Knowledge Property). A function q fulfills the *full knowledge* property if, for each agent that does not know at least one input value of another agent, any output in the range of q is *equally* possible. Formally, for any $1 \leq j \leq m$, fix $(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_m)$ and denote $z_y = |\{x_j | q(x_1, \ldots, x_j, \ldots, x_m) = y\}|$. A function q fulfills the *full knowledge* property if, for any possible output y in the range of q, z_y is the same[5].

We assume that each agent a prefers a certain output value p_a.

2.2.1 2-Knowledge Sharing

The 2-Knowledge Sharing problem is a Knowledge Sharing problem with exactly 2 distinct possible output values.

2.3 Coloring

In the Coloring problem [17, 28], Θ_L is any O such that $\forall a : o_a \neq \bot$ and $\forall (a, b) \in E : o_a \neq o_b$. We assume that every agent a prefers a specific color p_a.

3 Impossibility With No Knowledge

Here we prove that the common assumption that n is known is the key to the possibility of equilibrium for many problems: Without any a-priori knowledge about n, neither Knowledge Sharing nor Coloring have equilibria.

Let a be a malicious agent with δ outgoing edges. A possible deviation for a is to simulate imaginary agents a_1, a_2 and to answer over some of its edges as a_1, and over the others as a_2, as illustrated in Figure 1. From this point on a acts as if it is 2 agents. Here we assume that the *id* space is much larger than n, allowing us to disregard the probability that the fake *id* collides with an existing *id*, an issue dealt with in [11].

In our proofs we consider a weakened cheating agent that must decide on its duplication scheme at the very beginning of the algorithm, before any messages are exchanged. Thus, when the algorithm begins, it runs in a modified graph G' that is not the true graph G and contains duplications, but cannot be altered further by a cheater during the run of the algorithm. If this weakened cheater contradicts the possibility of equilibria, then surely a cheater that can make additional duplications while the algorithm runs would be able to

[5] The definition assumes input values are drawn uniformly, otherwise the definition of z_y can be expanded to the sum of probabilities over every input value for x_j.

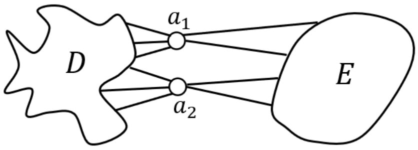

Figure 2 Graph H created by two arbitrary sub-graphs D,E.

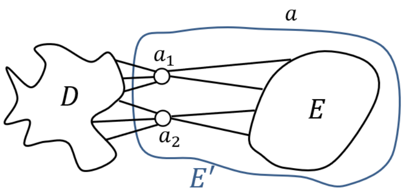

Figure 3 Example of agent a pretending to be $E' = E \cup \{a_1, a_2\}$.

adapt to the information it receives and increase its utility by creating more duplications[6]. This weakening only strengthens our impossibility proofs.

Regarding the output vector, notice that an agent that pretends to be more than one agent still outputs a *single* output at the end. The duplication causes agents to execute the algorithm as if it is executed on a graph G' (with the duplicated agents) instead of the original graph G; however, the output is considered legal if $O = (o_a, o_b, \dots) \in \Theta_L$ rather than if $(o_{a_1}, o_{a_2}, o_b, \dots) \in \Theta_L$.

It is important to emphasize that for any non-trivial distributed algorithm that is an equilibrium, the outcome cannot be calculated using only private data without communication. For rational agents, no agent can calculate the output privately at the beginning of the algorithm, since if it could calculate the output and know that its resulting utility will be 0, it would surely lie over its initial information to avoid losing, preventing equilibrium. If it knows its resulting utility is 1, it has no incentive to cheat. But there isn't always a solution in which everyone gains. This means that at round 0, for any agent a and any step s of the agent that does not necessarily result in algorithm failure, it must hold that: $\mathbb{E}_{s,0}[u_a] \notin \{0, 1\}$ (a value of 0 means an agent will surely not get its preference, and 1 means it is guaranteed to get its preference).

In this section we label agents in the graph as $a_1, ..., a_n$, set in a clockwise manner in a ring and in an arbitrary order in any other topology. These labels are not known to the agents themselves.

3.1 Impossibility of Knowledge Sharing

▶ **Theorem 5.** *There is no algorithm for Knowledge Sharing that is an equilibrium in a 2-connected graph when agents have no a-priori knowledge of n.*

Proof. Assume by contradiction that Λ is a Knowledge Sharing algorithm that is an equilibrium in any graph of agents who have absolutely no knowledge about n. Let D, E be two arbitrary 2-connected graphs of rational agents. Consider the execution of Λ on graph H created by D, E, and adding two nodes a_1, a_2 and connecting these nodes to 1 or more arbitrary nodes in both D and E (see Figure 2).

Recall that the vector of agents' inputs is denoted by $I = i_1, i_2, \cdots, i_n$, and $n = |H| = |D| + |E| + 2$. Let t_D be the first round after which $q(I)$ can be calculated from the collective information that all agents in D have (regardless of the complexity of the computation), and

[6] A cheater can be forced to commit using a *Wake-Up* protocol. Since no mechanism exists to ensure authenticity, an agent will choose what information to send (false ID, false input, false neighbors). The exchanged information, as Theorem 5 shows, is already altered by a cheater and the process is *not* an equilibrium.

similarly t_E the first round after which $q(I)$ can be calculated in E. Consider the following three cases:

1. $\boldsymbol{t_E < t_D}$: $q(I)$ cannot yet be calculated in D at round t_E. Let $E' = E \cup \{a_1, a_2\}$. Since $E \subset E'$, the collective information in E' at round t_E is enough to calculate $q(I)$. Since n is not known, an agent a could emulate the behavior of E', making the agents believe the algorithm runs on H rather than D. In this case, this cheating agent knows at round t_E the value of $q(I)$ in this execution, but the collective information of agents in D is not enough to calculate $q(I)$, which means the output of agents in D still depends on messages from E', the cheater. Thus, if a learns that the output $q(I) \neq p_a$, it can simulate all possible runs of the algorithm in a state-tree, and select a course of action that has at least some probability of leading to an outcome $q(I) = p_a$. Such a message surely exists because otherwise, D would have also known the value of $q(I)$. In other words, a finds a set of messages that *might* cause the agents in D to decide a value $x \neq q(I)$. In the case where $p_a = x$, agent a increases its expected utility by sending a set of messages different than that decreed by the protocol. Thus, agent a has an incentive to deviate, contradicting distributed equilibrium.
2. $\boldsymbol{t_D = t_E}$: both E and D have enough collective information to calculate $q(I)$ at the same round. The collective information in E at round t_E already exists in E' at round $t_E - 1$. Since $t_D = t_E$, the collective information in D is not enough to calculate $q(I)$ in round $t_E - 1$. Thus, similarly to Case 1, a can emulate E' and has an incentive to deviate.
3. $\boldsymbol{t_E > t_D}$: Symmetric to Case 1.

Thus, Λ is not an equilibrium for the Knowledge Sharing problem. ◀

▶ **Corollary 6.** *When a cheating agent pretends to be more than n agents, there is no algorithm for Knowledge Sharing that is an equilibrium when agents have no a-priori knowledge of n.*

Proof. Let H be a graph such that $|D| = |E|$. Follow the proof of Theorem 5. ◀

3.2 Impossibility of Coloring

The proof of Theorem 5 relies on the Full Knowledge property of the Knowledge Sharing problem, i.e., no agent can calculate the output before knowing all the inputs. Recall that the Coloring problem, however, is a more local problem [30], and nodes may color themselves without knowing anything about distant nodes.

▶ **Theorem 7.** *There is no algorithm for Coloring that is an equilibrium in a 2-connected graph when agents have no a-priori knowledge of n.*

Proof. Our proof is constructed by showing a type of graph in which a cheater could deviate to increase its expected utility, regardless of the algorithm. Surprisingly, this graph is simply a *ring*. Recall that an agent outputs a *single* color, even if it pretends to be several agents. In Coloring, a cheating agent only wishes to influence the output color of its *original neighbors* to enable it to output its preferred color while maintaining the legality of the output. The key to showing an incentive to deviate is defining a way to assess the precise point in which a cheater gains an advantage. We do this by generalizing the notion of *expected utility*:

▶ **Definition 8** (Group Expected Utility)**.** Let r be a round in an execution ε, and let M be a group of agents. For any set $S = \{s_1, \ldots, s_{|M|}\}$ of steps of agents in M, let Ψ be the set of all possible executions for which the same messages traverse the links that income and outgo to/from M as in ε until round r, and in round r each agent in M takes the corresponding step in S. For each possible output vector O, let x_O be the sum of probabilities over Ψ that

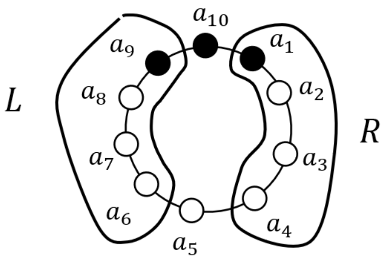

Figure 4 Ring with 3 agents a_9, a_{10}, a_1 colliding on their preferred color.

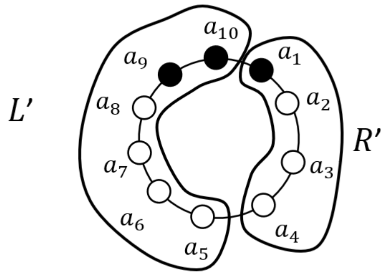

Figure 5 Ring with 3 agents colliding on their preferred color, with groups L', R'.

O is decided by the protocol. For any agent v, the *Group Expected Utility* of u_v by M taking steps S at round r in execution ε is: $\mathbb{E}_{M,S,r}[u_v] = \sum\limits_{O \in \Theta} x_O u_v(O)$.

Note that agents can also estimate the expected utility of *other* agents by considering a different utility function over the same output vectors of the execution ε.

Assume by contradiction that Γ is a Coloring algorithm that is an equilibrium in a ring with n agents $\{a_1, \ldots, a_n\}$. Let G be a ring with a segment of k consecutive agents, $k \geq 3$, all of which have the same color preference p. Assume w.l.o.g., they are centered around a_n if k is odd and around a_n, a_1 if even. Let L be the group of agents $\{a_{n-1}, \ldots, a_{\lfloor \frac{n}{2} \rfloor + 1}\}$, and R the group of agents $\{a_1, \ldots, a_{\lceil \frac{n}{2} \rceil - 1}\}$. Denote $L' = L \cup \{a_{\lceil \frac{n}{2} \rceil}, a_n\}$ and $R' = V \setminus L'$ (see Figures 4 and 5).

▶ **Definition 9.** Let Y be a group of agents (e.g., L or R). In any round r in an execution, let $S^r(Y)$ denote the vector of steps of agents in Y according to the protocol. We say Y *knows* the utility of agent a if it holds that $\mathbb{E}_{Y,S^r(Y)}[u_a] \in \{0, 1\}$. We say Y *does not know* the utility of agent a if $0 < \mathbb{E}_{Y,S^r(Y)}[u_a] < 1$.

Recall that at round 0 no agent (or group of agents) knows its utility or the utility of any other agent. Consider an execution of Γ on ring G and the groups L, R in the following cases:

1. L does not know u_{a_n} throughout the entire execution of the algorithm, i.e., for agents in L it holds that $0 < Pr[o_n \neq p] < 1$. Then if L is emulated by a cheating agent, it has an incentive to deviate and set its output to p (as otherwise its utility is guaranteed to be 0).
2. L knows u_{a_n} at some round t_L, and R does not know u_{a_n} before round t_L. Consider round $t_L - 1$ and group L': In round t_L, L knows the utility of a_n, thus the collective information of agents in L at round t_L already exists in L' at round $t_L - 1$. If L' knows that $u_{a_n} = 1$, then its utility is already 1; otherwise, L' knows that $u_{a_n} = 0$. Consider the group $R' \subset R$, that does not know u_{a_n} at round $t_L - 1$. If L' is emulated by a cheating agent a, it can send messages that increase its probability to output p from 0 to some positive probability, increasing its expected utility and thus it has an incentive to deviate.
3. R knows u_{a_n} before round t_L: symmetric to Case 2.

By the contradictory example for a ring, there is no equilibrium for Coloring 2-connected graphs when agents have no a-priori knowledge of n. ◀

4 Algorithms

Here we present algorithms for Knowledge Sharing (Section 4.1) and Coloring (Section 4.2). In the previous section we saw that in Knowledge Sharing, if a duplicating agent can pretend to be more than n agents equilibrium is impossible (Corollary 6). The Knowledge Sharing algorithm presented here is an equilibrium in a ring when no cheating agent pretends to be more than n agents, proving a tight bound and improving the Knowledge Sharing algorithm in [7]. The Coloring algorithm is an equilibrium in any 2-connected graph when agents a-priori know n.

Notice that using an algorithm as a subroutine is not trivial in this setting, even if the algorithm is an equilibrium, as the new context as a subroutine may allow agents to deviate towards a different objective than was originally proven. Thus, whenever a subroutine is used, its equilibrium should be proved.

4.1 Knowledge Sharing in a Ring

First we describe the `Secret-Transmit`(i_a,r,b) building block in which an agent a transmits its input i_a to an agent b of its choosing, such that b learns i_a at round r and no other agent in the ring learns any information about this input. Several `Secret-Transmits` can be executed concurrently. To achieve this, agent a selects a random number R_a, and let $X_a = R_a \oplus i_a$. It sends R_a clockwise and X_a counter-clockwise until each reaches the agent before b. At round $r-1$, these neighbors of b simultaneously send b the values X_a and R_a, thus b receives the information at round r.

We assume a global orientation around the ring. This assumption can be easily relaxed via Leader Election [7], which is an equilibrium in this application since the orientation has no effect on the output. The algorithm works as follows:

Algorithm 1 Knowledge Sharing in a Ring.

1: All agents execute Wake-Up [7] to learn the ids of all agents and n', the size of the ring (which may include duplications)
2: For each agent a, denote b_a^1 the clockwise neighbor of a, and b_a^2 the agent at distance $\lfloor \frac{n'}{2} \rfloor$ counter-clockwise from a
3: Each agent a simultaneously performs:
`SecretTransmit`(i_a, n', b_a^1)
`SecretTransmit`(i_a, n', b_a^2)
4: At round $n'+1$, each agent sends its input around the ring
5: At round $2n'$ output $q(I)$

▶ **Theorem 10.** *In a ring, Algorithm 1 is an equilibrium when no cheating agent pretends to be more than n agents.*

Proof. Assume by contradiction that a cheating agent pretending to be $d \leq n$ agents has an incentive to deviate. W.l.o.g., the duplicated agents are $a_1, \ldots, a_d$ (recall the indices $1, \ldots, n'$ are not known to the agents).

Let n' be the size of the ring including the duplicated agents, i.e., $n' = n + d - 1$. The clockwise neighbor of $a_{n'}$ is a_1, denoted $b^1_{a_{n'}}$. Denote $a_c = b^2_{a_{n'}}$ the agent at distance $\lfloor \frac{n'}{2} \rfloor$ counter-clockwise from $a_{n'}$, and note that $c \geq d$.

When $a_{n'}$ calls `Secret-Transmit` to a_1, $a_{n'}$ holds $R_{n'}$ of that transmission until round $n'-1$. When $a_{n'}$ calls `Secret-Transmit` to a_c, a_{c+1} holds $X_{n'}$ of that transmission until round $n' - 1$. By our assumption, the cheating agent duplicated into $a_1, \ldots, a_d$. Since $d < c + 1$, the cheater receives at most one piece ($X_{n'}$ or $R_{n'}$) of each of $a_{n'}$'s transmissions before round n'. So, there is at least one input that the cheater does not learn before round n'. According to the Full Knowledge property (Definition 4), for the cheater at round $n' - 1$ any output is equally possible, so its expected utility for any value it sends is the same, thus it has no incentive to cheat regarding the values it sends in round $n' - 1$.

Let $a_j \in \{a_1, \ldots, a_d\}$ be an arbitrary duplicated agent. In round n', i_{a_j} is known by its clockwise neighbor $b^1_{a_j}$ and by $b^2_{a_j}$, the agent at distance $\lfloor \frac{n'}{2} \rfloor$ counter-clockwise from a_j. Since the number of counter-clockwise consecutive agents in $\{b^1_{a_j}, a_j, \ldots, b^2_{a_j}\}$ is greater than $\lceil \frac{n'}{2} \rceil \geq n$, at least one of $b^1_{a_j}, b^2_{a_j}$ is not a duplicated agent. Thus, at round n', the input of each agent in $\{a_1, \ldots, a_d\}$ is already known by at least one agent $\notin \{a_1, \ldots, a_d\}$.

At round $n' - 1$ the cheater does not know the input value of at least one other agent, so by the Full Knowledge property it has no incentive to deviate. At round n' for each duplicated agent, its input is already known by a non-duplicated agent, which disables the cheater from lying about its input from round n' and on.

Thus, the cheating agent has no incentive to deviate, contradicting our assumption. ◀

In other words, in Algorithm 1 an agent has no incentive to deviate unless it duplicates more than n agents.

4.2 Coloring in General Graphs

Here, agents are given exact a-priori knowledge of n. Since agent *ids* are private and agents may cheat about their *id*, *ids* cannot be used to decide which of two neighbors that desire the same color actually gets it. However, an *orientation* over an edge is shared by both agents, and an acyclic orientation over the graph can be used to break ties.

Note that since the agents are rational, unless agent a knows that one or more of its neighbors output a's preferred color p_a, it will output p_a itself, regardless of the algorithm step, which is a *deviation*. Thus, any coloring algorithm must ensure that whenever an agent can output its preferred color, it does, otherwise the agent has a trivial incentive to deviate.

We present Algorithm 2 that uses two subroutines to obtain a coloring. *Draw* (Algorithm 3) is an equilibrium in which agent a randomizes a number different from those of its neighbors and commits to it. *Prompt* (Algorithm 4) is a query that ensures a receives the correct drawn number from a neighbor. A full explanation is provided in the full paper [6].

▶ **Theorem 11.** *Algorithm 2 is an equilibrium for Coloring when agents a-priori know n.*

Proof. Let a be an arbitrary agent. Assume in contradiction that at some round r there is a possible cheating step s such that $s \neq s_r$ and $\mathbb{E}_{s,r}[u_a] > \mathbb{E}_{s_r,r}[u_a]$.

Consider the possible deviations for a in every phase of Algorithm 2:

- Wake-Up: The order by which agents initiate Algorithm 3 has no effect on the order by which they will later set their colors. Hence, a has no incentive to publish a false *id* in the Wake-Up building block.
- *Draw* is an equilibrium: An agent and a witness send a random number simultaneously.
- Publishing a false S value will be caught by the verification in step 10 of Algorithm 2.
- Sending a color message not in order will be immediately recognized by the neighbors, since S values were verified.
- Agent a might output a different color than the color dictated by Algorithm 2. But if the preferred color is available, then outputting it is the only rational behavior. Otherwise, the utility for the agent is already 0 in any case. ◀

Algorithm 2 Coloring via Acyclic Orientation (for agent a).

1: Run Wake-Up ▷ After which all agents know graph topology
2: set $T := \varnothing$ ▷ T is the set of values already taken by a's neighbors ($N(a)$)
3: **for** $i = 1, ..., n$ **do**
4:     **if** $id_a = i$'th largest id in V **then** ▷ Draw random numbers in order of id
5:         $Draw(T)$
6:     **else**
7:         **wait** $|Draw|$ rounds ▷ Wait for $Draw$, takes a constant number of rounds
8:         **if** received $S(v)$ from $v \in N(a)$ **then** ▷ $S(v)$ is the value of v from $Draw$
9:             $T = T \cup \{S(v)\}$ ▷ Add $S(v)$ to set of taken values
10: **for** $u \in N(a)$ **simultaneously do**
11:     $Prompt(u)$ ▷ Since we must validate the value received in line 8
12: **wait** until all prompts are completed in the entire graph ▷ At most n rounds
13: **for** round $t = 1, ..., n$ **do**:
14:     **if** $S(a) = t$ **then** ▷ Wait for your turn, decreed by your S value
15:         **if** $\forall v \in N(a) : o_v \neq p_a$ **then** $o_a := p_a$
16:         **else** $o_a :=$ minimum color unused by any $v \in N(a)$
17:         **send** o_a to $N(a)$

Algorithm 3 Draw(T) Subroutine (for agent a and the witness $w(a)$).

Denote $X = \{1, ..., n\} \setminus T$ ▷ X is the set of numbers not drawn by neighbors
1: $w(a) :=$ node b s.t. id_b is minimal in N(a) ▷ $N(a)$ is the set of neighbors of a
send *witness* to $w(a)$ ▷ choose neighbor with minimal id as witness
2: $r(a) := random\{1, ..., |X|\}$ drawn by a
$r(w(a)) := random\{1, ..., |X|\}$ drawn by $w(a)$
send $r(a)$ to $w(a)$
receive $r(w(a))$ from $w(a)$ ▷ a and witness jointly draw a random number
3: Let $q := r(a) + r(w(a))$ mod $|X|$.
Set $S(a) := q$'th largest number in X
send $S(a)$ to all $u \in N(a)$ ▷ Calculate $S(a)$ and publish to neighbors

Algorithm 4 Prompt(b) Subroutine (for agent a).

upon receiving a $prompt(b)$ message from $b \in N(a)$:
1: $p :=$ shortest simple path $a \to w(a) \to b$ ▷ $w(a)$ is set by a preceding call to `Draw`
send $S(a), b$ via p ▷ If $v \neq w(a)$ is asked to relay $S(a)$, v fails the algorithm
send $S(a)$ to b via $e = (a, u)$ ▷ b validates that both messages received are consistent

Table 2 Knowledge Bounds; summary of results.
$*$ – Bound is tight only in rings.

Bound	Problem
$\alpha + 1$	Leader Election
$2\alpha - 2$	Knowledge Sharing$*$
∞	Coloring$*$, 2-Knowledge Sharing
unbounded	Partition, Orientation

5 How Much Knowledge Is Necessary?

In Section 3 we have shown that with rational agents, knowledge of n is crucial; however, in some cases, *bounds* on the value of n may be enough for equilibrium. In this section we examine the effects of a-priori knowledge that *bound* the possible value of n. We show that the possibility of equilibria depends on the range $[\alpha, \beta]$ in which n might be, and show these ranges for different problems. Table 2 summarizes our results.

Partition and Orientation have equilibria without *any* knowledge of n; however, the former is constrained to even-sized rings, and the latter is a trivial problem in distributed computing (radius 1 in the *LOCAL* model [29]).

▶ **Definition 12** ((α, β)-Knowledge)**.** We say agents have (α, β)-*Knowledge* about the actual number of agents n, $\alpha \leq \beta$, if all agents a-priori know that the value of n is in $[\alpha, \beta]$. Agents have no information about the distribution over $[\alpha, \beta]$, i.e., they assume it is uniform.

▶ **Definition 13** (f-Bound)**.** Let $f : \mathbb{N} \to \mathbb{N}$. A problem $\mathbb{P}$ is *f-bound* if:

- There exists an algorithm for $\mathbb{P}$ that is an equilibrium given (α, β)-Knowledge for any α, β such that $\beta \leq f(\alpha)$.
- For any algorithm for $\mathbb{P}$, there exist α, β where $\beta > f(\alpha)$ such that given (α, β)-Knowledge the algorithm is not an equilibrium.

In other words, a problem is f-bound if given (α, β)-Knowledge, there is an equilibrium when $\beta \leq f(\alpha)$, and there is no equilibrium if $\beta > f(\alpha)$.

A problem is ∞-bound if there is an equilibrium given *any* bound f, but there is no equilibrium with $(1, \infty)$-Knowledge. A problem is *unbounded* if there is an equilibrium with $(1, \infty)$-Knowledge.

Consider an agent a at the start of a protocol given (α, β)-Knowledge. If a pretends to be a group of d agents, it can be caught when $d + n - 1 > \beta$, since agents might count the number of agents and catch the cheater. Moreover, *any* duplication now involves some risk since the actual value of n is not known to the cheater (similar to [11]).

An arbitrary cheating agent a simulates executions of the algorithm for every possible duplication, and evaluates its expected utility. Denote D a duplication scheme in which an agent pretends to be d agents. Let $P_D = P[d + n - 1 \leq \beta]$ be the probability, from agent a's perspective, that the overall size does not exceed β. If for agent a there exists a duplication scheme D at round 0 such that $\mathbb{E}_{D,0}[u_a] \cdot P_D > \mathbb{E}_{s(0),0}[u_a]$, then agent a has an incentive to deviate and duplicate itself. For each problem we look for the maximal range of α, β where no d exists that satisfies the inequality above.

5.1 Knowledge Sharing

▶ **Theorem 14.** *Knowledge Sharing in a ring is* $(2\alpha - 2)$*-bound.*

Proof. Assume agents have (α, β)-knowledge for some α, β. A cheating agent a chooses d, the number of agents it pretends to be, that maximizes its expected utility.

Let k be the size of the range of the output function q (Definition 4). By Definition 4, any output is *equally* possible. Therefore, without deviation the expected utility of a at round 0 is: $\mathbb{E}_{s(0),0}[u_a] = \frac{1}{k}$.

Corollary 6 shows that when a cheating agent pretends to be *more* than n agents, it gains an advantage (thus there is no equilibrium). According to Theorem 10, Algorithm 1 is an equilibrium for Knowledge Sharing in a ring when a cheating agent pretends to be n agents or less. If n is in the range $[\alpha, \beta]$, a duplicates to d agents to maximize the probability that $d > n$ and thus the duplication increases its expected utility, while also minimizing the probability that $d + n - 1 > \beta$ and a is caught.

To successfully gain an advantage a must duplicate to at least $d \geq \alpha$, or otherwise d is surely $< n$ and by Theorem 10, there is an equilibrium. Further notice that $d \leq \lceil \frac{\beta}{2} \rceil + 1$ (the $+1$ is a itself) since a higher value of d increases the probability of a to be caught without increasing the probability of gaining any advantage.

From a's perspective at the beginning of the algorithm, the value of n is uniformly distributed over $[\alpha, \beta]$. Let $X > \frac{1}{k}$ be the utility a gains by pretending to be $d > n$ agents if it is not caught, i.e., if $d + n - 1 \leq \beta$. The probability to duplicate to $d > n$ agents and not be caught is $\frac{d-\alpha}{\beta-\alpha+1}$. On the other hand, when pretending to be $d \leq n$ agents without being caught the utility of a does not change and is $\frac{1}{k}$, and this has a probability of $\frac{\lceil \frac{\beta}{2} \rceil + 1 - d}{\beta - \alpha + 1}$. In all other cases $d + n - 1 > \beta$ and a is caught, resulting in a utility of 0. Thus, the expected utility of agent a at round 0 is:

$$\mathbb{E}_{D,0}[u_a] = X \cdot \frac{d - \alpha}{\beta - \alpha + 1} + \frac{1}{k} \cdot \frac{\lceil \frac{\beta}{2} \rceil + 1 - d}{\beta - \alpha + 1} \tag{2}$$

The expected utility in (2) reaches a maximum at $d = \lfloor \frac{\beta}{2} \rfloor + 1$, so set d to that number as the best cheating strategy. Recall that a deviates from the algorithm whenever $\mathbb{E}_{D,0}[u_a] > \frac{1}{k}$:

$$\mathbb{E}_{D,0}[u_a] = X \cdot \frac{\lfloor \frac{\beta}{2} \rfloor + 1 - \alpha}{\beta - \alpha + 1} + \frac{1}{k} \cdot \frac{\lceil \frac{\beta}{2} \rceil - \lfloor \frac{\beta}{2} \rfloor}{\beta - \alpha + 1} > \frac{1}{k} \tag{3}$$

As k grows, $\frac{1}{k}$ approaches 0. By setting $\frac{1}{k} = 0$ Equation 3 shows that agent a has an incentive to deviate when $\lfloor \frac{\beta}{2} \rfloor + 1 - \alpha > 0$. When β is even: $\beta > 2\alpha - 2$, otherwise: $\beta > 2\alpha - 1$. Thus, Algorithm 1 is an equilibrium for Knowledge Sharing when agents have (α, β)-knowledge such that $\beta \leq 2\alpha - 2$, and there exist $\alpha, \beta > 2\alpha - 2$ such that there is no equilibrium for Knowledge Sharing when agents have (α, β)-knowledge. By Definition 13, Knowledge Sharing is $(2\alpha - 2)$-bound in rings. ◀

To find the f-bound for any specific value of k and in any graph, we derive β as a function of α:

$$\begin{cases} \beta \text{ is even} & \beta(kX - 2) > 2\alpha kX - 2kX - 2\alpha + 2 \\ \beta \text{ is odd} & \beta(kX - 2) > 2\alpha kX - kX - 2\alpha \end{cases} \tag{4}$$

▶ **Corollary 15.** *2-Knowledge Sharing in a ring is ∞-bound.*

Proof. The inequalities in 4 are satisfiable only if $X > 2 \cdot \frac{1}{k}$. Since $X \leq 1$, the inequalities cannot be satisfied in 2-Knowledge Sharing ($k = 2$) and a has no incentive to deviate, given *any* bound on n. ◀

Algorithm 5 Coloring in a Ring.

1: Wake-Up to learn the size of the ring.
2: Assume arbitrary global direction over the ring (can be relaxed via Leader Election [7]).
3: Run 2-Knowledge Sharing to randomize a single global bit $b \in \{0, 1\}$.
4: Publish the preferred color of each agent simultaneously over the entire ring.
5: In each group of consecutive agents that prefer the same color, if $b = 0$ the even agents (according to the orientation) output their preferred color, else the odd agents do.
6: If an agent has no neighbors who prefer the same color, it outputs its preferred color.
7: Any other agent outputs the minimal available color.

5.2 Coloring

▶ **Theorem 16.** *Coloring in a ring is ∞-bound.*

Proof. Consider Algorithm 5 which solves coloring in a ring using 2-Knowledge Sharing.

It is easy to see that Algorithm 5 is an equilibrium and results in a legal coloring of the ring. It uses 2-Knowledge Sharing and thus, following Corollary 15, it proves Theorem 16. ◀

5.3 Leader Election

In the Leader Election problem, each agent a outputs $o_a \in \{0, 1\}$, where $o_a = 1$ means that a was elected leader and $o_a = 0$ means otherwise. $\Theta_L = \{O | \exists a : o_a = 1, \forall b \neq a : o_b = 0\}$. An agent prefers either 0 or 1.

▶ **Theorem 17.** *Leader Election is $(\alpha + 1)$-bound.*

Proof. Recall that any Leader Election algorithm must be *fair* [4], i.e., every agent must have equal probability of being elected leader for the algorithm to be an equilibrium.

Given $f(\alpha) = \alpha+1$, the actual number of agents n is either α or $\alpha+1$. If an agent follows the protocol, the probability of being elected is $\frac{1}{n}$. If it duplicates itself once, the probability that one of its instances is elected is $\frac{2}{n+1}$, but if $n = \alpha + 1$ then $n' > \beta$, it is easily detected and its utility is 0. Thus $\mathbb{E}_{D,0}[u_a] = \frac{1}{2}\frac{2}{n+1} < \frac{1}{n}$, i.e., no agent has an incentive to deviate.

Given $f(\alpha) = \alpha + 2$, then n is in $[\alpha, \alpha + 2]$. If an agent follows the protocol, its expected utility is still $\frac{1}{n}$. If it duplicates itself once, the probability that a duplicate is elected is still $\frac{2}{n+1}$, however *only* if $n = \alpha + 2$ then $n' > \beta$ and the cheater is caught. Thus, $\mathbb{E}_{D,0}[u_a] = \frac{2}{3}\frac{2}{n+1} > \frac{1}{n}$ for any $n > 3$. So the agent *has* an incentive to deviate. ◀

5.4 Ring Partition

In the Ring Partition problem, the agents of an even-sized ring are partitioned into two, equally-sized groups: group 0 and group 1. An agent prefers to belong to either group 0 or 1. In the full paper [6] we prove:

▶ **Theorem 18.** *Ring Partition is unbounded.*

5.5 Orientation

In the Orientation problem, the two ends of each edge must agree on a direction for the edge. An agent prefers certain directions for its edges. In the full paper [6] we prove:

▶ **Theorem 19.** *The Orientation problem is unbounded.*

6 Discussion

In this paper we have shown that the assumption that n is a-priori known, commonly made in previous works, has a critical role in the possibility of equilibrium. In realistic scenarios, the exact size of the network may not be known to all members, or only estimates on the exact size are known in advance. In such networks, the use of duplication gives an agent power to affect the outcome of most algorithms, and in some cases makes equilibrium impossible. In this work we did not identify any problem that requires exact knowledge of n for equilibrium. Even in Leader Election, equilibrium is possible as long as n is known to be in a margin of 2.

When bounds on n are known, the f-bounds we have proven in Section 5 show that the initial knowledge required for equilibrium depends on the balance between two factors: The amount of duplications necessary to increase an agent's expected utility, and the probability that the cheater is caught duplicating. In order for an agent to have an incentive to duplicate itself, an undetected duplication needs to be more profitable than following the algorithm while also involving low risk of being caught.

Our results suggest several open directions that may be of interest:

1. Finding an equilibrium for Knowledge Sharing in a general graph with at most n duplications. This would be the missing piece that, along with our impossibility proof in Theorem 5, would prove the f-bound of $2\alpha - 2$ is tight for general graphs.
2. Algorithms and impossibility results for other problems, as well as tight f-bounds.
3. Finding a problem that is α-bound, i.e., has an equilibrium only when n is known exactly.
4. Finding more problems that have equilibrium without any knowledge of n in any graph (unlike Partition) and a non-constant radius in the LOCAL model (unlike Orientation).
5. Exploring the effects of initial knowledge of n in an asynchronous setting.

References

1 Ittai Abraham, Lorenzo Alvisi, and Joseph Y. Halpern. Distributed computing meets game theory: Combining insights from two fields. *SIGACT News*, 42(2):69–76, 2011. doi:10.1145/1998037.1998055.

2 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 *PODC*, pages 53–62, 2006. doi:10.1145/1146381.1146393.

3 Ittai Abraham, Danny Dolev, and Joseph Y. Halpern. Lower bounds on implementing robust and resilient mediators. In *TCC*, pages 302–319, 2008. doi:10.1007/978-3-540-78524-8_17.

4 Ittai Abraham, Danny Dolev, and Joseph Y. Halpern. Distributed protocols for leader election: A game-theoretic perspective. In *DISC*, pages 61–75, 2013. doi:10.1007/978-3-642-41527-2_5.

5 Norman Abramson. The aloha system: Another alternative for computer communications. In *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*, AFIPS '70 (Fall), pages 281–285, New York, NY, USA, 1970. ACM.

6 Y. Afek, S. Rafaeli, and M. Sulamy. Cheating by Duplication: Equilibrium Requires Global Knowledge. *ArXiv e-prints*, 2017. arXiv:1711.04728.

7 Yehuda Afek, Yehonatan Ginzberg, Shir Landau Feibish, and Moshe Sulamy. Distributed computing building blocks for rational agents. In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*, PODC '14, 2014.

8 Amitanand S. Aiyer, Lorenzo Alvisi, Allen Clement, Michael Dahlin, Jean-Philippe Martin, and Carl Porth. Bar fault tolerance for cooperative services. In *SOSP*, pages 45–58, 2005. doi:10.1145/1095810.1095816.

9 Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, 2004.

10 B. Awerbuch, M. Luby, A. V. Goldberg, and S. A. Plotkin. Network decomposition and locality in distributed computation. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, pages 364–369, Washington, DC, USA, 1989. IEEE Computer Society. `doi:10.1109/SFCS.1989.63504`.

11 D. Bank, M. Sulamy, and E. Waserman. Reaching Distributed Equilibrium with Limited ID Space. *ArXiv e-prints*, 2018. `arXiv:1804.06197`.

12 Imre Bárány. Fair distribution protocols or how the players replace fortune. *Math. Oper. Res.*, 17(2):327–340, 1992. `doi:10.1287/moor.17.2.327`.

13 Elchanan Ben-Porath. Cheap talk in games with incomplete information. *J. Economic Theory*, 108(1):45–71, 2003. `doi:10.1016/S0022-0531(02)00011-X`.

14 Rajat Bhattacharjee and Ashish Goel. Avoiding ballot stuffing in ebay-like reputation systems. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-peer Systems*, P2PECON '05, pages 133–137, New York, NY, USA, 2005. ACM.

15 Monica Bianchini, Marco Gori, and Franco Scarselli. Inside pagerank. *ACM Trans. Internet Technol.*, 5(1):92–128, 2005.

16 Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-peer Systems*, P2PECON '05, pages 128–132, New York, NY, USA, 2005. ACM.

17 Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf. Control*, 70(1):32–53, 1986.

18 Varsha Dani, Mahnush Movahedi, Yamel Rodriguez, and Jared Saia. Scalable rational secret sharing. In *PODC*, pages 187–196, 2011. `doi:10.1145/1993806.1993833`.

19 Yevgeniy Dodis, Shai Halevi, and Tal Rabin. A cryptographic solution to a game theoretic problem. In *CRYPTO*, pages 112–130, 2000. `doi:10.1007/3-540-44598-6_7`.

20 John R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag.

21 Georg Fuchsbauer, Jonathan Katz, and David Naccache. Efficient rational secret sharing in standard communication networks. In *TCC*, pages 419–436, 2010. `doi:10.1007/978-3-642-11799-2_25`.

22 S. Dov Gordon and Jonathan Katz. Rational secret sharing, revisited. In *SCN*, pages 229–241, 2006. `doi:10.1007/11832072_16`.

23 Adam Groce, Jonathan Katz, Aishwarya Thiruvengadam, and Vassilis Zikas. Byzantine agreement with a rational adversary. In *ICALP (2)*, pages 561–572, 2012. `doi:10.1007/978-3-642-31585-5_50`.

24 Joseph Y. Halpern and Xavier Vilaça. Rational consensus: Extended abstract. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 137–146, New York, NY, USA, 2016. ACM.

25 L. Kleinrock and F. Tobagi. Packet switching in radio channels: Part i - carrier sense multiple-access modes and their throughput-delay characteristics. *IEEE Transactions on Communications*, 23(12):1400–1416, December 1975.

26 Fabian Kuhn and Rogert Wattenhofer. On the complexity of distributed graph coloring. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '06, pages 7–15, New York, NY, USA, 2006. ACM. `doi:10.1145/1146381.1146387`.

27 Matt Lepinski, Silvio Micali, Chris Peikert, and Abhi Shelat. Completely fair sfe and coalition-safe cheap talk. In *PODC*, pages 1–10, 2004. `doi:10.1145/1011767.1011769`.

28 N. Linial. Legal coloring of graphs. *Combinatorica*, 6(1):49–54, 1986. doi:10.1007/BF02579408.

29 Nathan Linial. Distributive graph algorithms global solutions from local data. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, SFCS '87, pages 331–335, Washington, DC, USA, 1987. IEEE Computer Society. doi:10.1109/SFCS.1987.20.

30 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.

31 Anna Lysyanskaya and Nikos Triandopoulos. Rationality and adversarial behavior in multi-party computation. In *CRYPTO*, pages 180–197, 2006. doi:10.1007/11818175_11.

32 Robert McGrew, Ryan Porter, and Yoav Shoham. Towards a general theory of non-cooperative computation. In *TARK*, pages 59–71, 2003. doi:10.1145/846241.846249.

33 Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. When selfish meets evil: byzantine players in a virus inoculation game. In *PODC*, pages 35–44, 2006. doi:10.1145/1146381.1146391.

34 Rafael Pass and Elaine Shi. Hybrid Consensus: Efficient Consensus in the Permissionless Model. In *31st International Symposium on Distributed Computing (DISC 2017)*, 2017.

35 Rafael Pass and Elaine Shi. The sleepy model of consensus. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 380–409, Cham, 2017. Springer International Publishing.

36 Rafael Pass and Elaine Shi. Rethinking large-scale consensus. Cryptology ePrint Archive, Report 2018/302, 2018. URL: https://eprint.iacr.org/2018/302.

37 Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979. doi:10.1145/359168.359176.

38 Yoav Shoham and Moshe Tennenholtz. Non-cooperative computation: Boolean functions with correctness and exclusivity. *Theoretical Computer Science*, 343(1–2):97–113, 2005.

39 Márió Szegedy and Sundar Vishwanathan. Locality based graph coloring. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 201–207, New York, NY, USA, 1993. ACM. doi:10.1145/167088.167156.

40 Amparo Urbano and Jose E. Vila. Computational complexity and communication: Coordination in two-player games. *Econometrica*, 70(5):1893–1927, September 2002. URL: http://ideas.repec.org/a/ecm/emetrp/v70y2002i5p1893-1927.html.

41 Amparo Urbano and José E. Vila. Computationally restricted unmediated talk under incomplete information. *Economic theory*, 2004.

42 Edmund L. Wong, Isaac Levy, Lorenzo Alvisi, Allen Clement, and Michael Dahlin. Regret freedom isn't free. In *OPODIS*, pages 80–95, 2011. doi:10.1007/978-3-642-25873-2_7.

43 Assaf Yifrach and Yishay Mansour. Fair leader election for rational agents in asynchronous rings and networks. In *PODC '18*, 2018. doi:10.1145/3212734.3212767.