Verifiable Crowd Computing: Coping with Bounded Rationality

Lu Dong[1], Miguel A. Mosteiro[1(✉)], and Shikha Singh[2]

[1] Pace University, New York, NY 10038, USA
{ld41349n,mmosteiro}@pace.edu
[2] Williams College, Williamstown, MA 01267, USA
shikha@cs.williams.edu

Abstract. In this paper we use the repeated-games framework to design and analyze a master-worker (MW) mechanism, where a master repeatedly outsources computational tasks to workers in exchange for payment. Previous work on MW models assume that all workers are *perfectly rational* and aim to maximize their expected utility. Perfect rationality is a strong behavioral assumption because it requires that all workers follow the prescribed equilibrium. Such a model may be unrealistic in a practical setting, where some agents are not perfectly rational and may deviate from the equilibrium for short-term gains. Since the correctness of these MW protocols relies on workers following the equilibrium strategies, they are not robust against such deviations.

We augment the game-theoretical MW model with the presence of such bounded-rational players or *deviators*. In particular, we show how to design a repeated-games based MW Verifiable Crowd Computing mechanism that incentivizes the rational workers (or *followers*) to effectively punish such deviators through the use of terminal payments. We show that the master can use terminal payments to obtain correct answers to computational problems even in this more general model. We supplement our theoretical results with simulations that show that our mechanism outperforms related approaches.

Keywords: Crowd computing · Master-worker computing · Internet computing · Verifiable computation outsourcing · Repeated games · Algorithmic game theory

1 Introduction

Due to the escalation of large data sets and the high cost of supercomputers, most computation today is not being performed locally, but rather outsourced either as Cloud Computing, Grid Computing, or to Internet-connected personal computers (which we refer to as Crowd Computing[1]). In a Crowd Computing platform, a client outsources computational tasks to several untrusted workers,

[1] The denomination Crowd Computing has been used for different systems. We define our model in Sect. 3.

M. Li and X. Sun (Eds.): IJTCS-FAW 2022, LNCS 13461, pp. 59–78, 2022.
https://doi.org/10.1007/978-3-031-20796-9_5

often in exchange for money. Crowd Computing has proven to be a powerful and cost-effective alternative to setting up an expensive computational infrastructure locally. However, it also brings up several implementation challenges, the most prominent one being: how can a client ensure that the computational tasks have been performed correctly by the untrusted workers?

In this work, we study Crowd Computing in the master-worker (MW) model where computational tasks are assigned by a centralized entity called *master* to agents called *workers*. Workers are expected to compute the task and return the result. All communication between master and workers occurs in parallel, that is, tasks are assigned and the results are returned simultaneously by all workers.

Practical examples of such Crowd Computing systems include SETI@home [42], Foldit [32,45], and Mechanical Turk [3]. Reliable participation in these systems is induced through various incentives. For example, in Foldit the motivation is to participate in a game, in Mechanical Turk workers participate for profit compelled by a reputation system, whereas in SETI@home workers are altruistic volunteers.

Game-theoretical MW Model. Various models of worker-behavior have been studied in the MW literature in distributed computing, e.g., the workers have been assumed to be either *malicious* or *honest* [20,21,23,34–36,44].

The game-theoretic MW model assumes that all workers are rational and choose their strategy to maximize their expected utility [13,14,21,22,47]. While such a model is ideal for capturing the economic incentives that are present in these systems, it also imposes strong behavioral assumptions on the workers. In particular, the players are assumed to be perfectly rational. That is, all players play the prescribed equilibrium strategy. In practice, some strategic agents in a Crowd Computing system may not follow the equilibrium strategy for any number of reasons: carelessness, limited rationality (they may not believe in the expected utility guarantees of a probabilistic mechanism), they may be skeptical of the empty threats sustainining the equilbrium, or they may distrust the master. We call such players who are not perfectly-rational and may not follow the prescribed equilibrium as *deviators*.

There is evidence that even in systems that provide incentives for computing correctly, such deviators exist [4,26,30]. Since the master obtains the correct answer to the computational tasks only when all workers follow the equilibrium prescribed by the chosen incentives, the presence of such bounded-rational players, or *deviators* jeopardizes the correctness guarantees of such systems.

In this paper, we provide a principled approach to designing MW mechanisms for Crowd Computing in the presence of such "bounded rational" deviators. We say these deviators have bounded rationality to distinguish them from purely malicious players because while they initially deviate from the equilibrium, once they are punished by their rational peers, they realize that the threats are credible, and their utility will be maximized by following the equilibrium. That is, we assume that after being punished deviators behave as followers.

Master-Worker Model with Deviators. In this work, we augment the repeated-games-based MW model of Fernández Anta et al. [24] in two ways.

First, we consider pools of workers that include *deviators*.[2] In particular, we assume that some players in the system are perfectly-rational in the game-theoretic sense and, hence, will follow the prescribed equilibrium (we call them *followers*) and the rest are *deviators*, who initially may not follow the equilibrium strategy. Second, in contrast to the *infinitely* repeated game studied in [24], we consider *finitely* repeated games. That is, the MW interaction lasts for predetermined number of rounds (with a possible extension as explained in Sect. 3).

The presence of deviators in the MW model poses several challenges. First, we need a mechanism to identify and appropriately punish such deviators. Second, in a repeated-games framework such a punishment must be levied by the followers, who incur a short-term utility loss in doing so. Thus, we need a mechanism to incentivize the followers to punish deviators on behalf of the master. We achieve this through the use of ***terminal payments***, which are additional payments provided to the followers at the end of a finitely-repeated game.

Terminal payments in finitely-repeated games has been studied by Gossner [27], who proved a Folk Theorem for such games when players use mixed-strategies. Gossner introduced these payments precisely to handle the presence of deviators. We follow Gossner's model of terminal payments and use them to compensate followers at the end of the MW protocol.

Finitely-repeated games and terminal payments are particularly appealing from a practical standpoint in Crowd Computing systems as they better capture the practice of (a) hiring workers for limited periods, and (b) keeping track of their behavior and rewarding compliance through payments, respectively. Extensions such as modeling different types of deviators, or collusion among workers are also interesting, and are left to future work.

1.1 Main Contributions

We summarize the main contributions of our paper below.

- *Modeling Deviators.* We generalize the repeated-games MW model to allow for the presence of bounded-rational workers or deviators. Bounded-rationality is a concept that has gained popularity in the computational applications of game theory (e.g. see. [5,10–12,16,28,29,43]) because it bridges the gap between game theory and rational behavior in practice. Our bounded-rationality models intentional deviations, as opposed to other models where deviations are accidental (e.g. trembling-hand equilibria [9]).
- *Terminal Payments.* We show how to implement terminal payments in a constructive way in the repeated-games model. This notion was previously only studied in an existential context. In particular, Gossner [27] showed that a mixed-strategy equilibrium of finitely-repeated games exists in the presence of terminal payments. In this paper, we construct a mechanism that admits such an equilibrium in the presence of deviators.

[2] The notion of deviators is only considered as part of the simulations in [24]; see Sects. 2 and 6 for further comparison.

- *Verifiable MW Computing in the Presence of Deviators.* As our main result (Theorem 1) we provide a MW mechanism that is robust against deviating workers—that is, the master obtains the correct answer in expectation and asymptotically almost surely under certain mild conditions. We achieve this through a finitely-repeated MW game and the use of terminal payments. Thus, our mechanism improves over the work of Fernández Anta et al. [24] in achieving correctness guarantees in a weaker behavioral model that includes bounded-rationality.
- *Simulation Results.* We also compare experimentally the performance of our mechanism to the previous work of Christoforou et al. [15] and Fernández-Anta et al. [24]. We compare these approaches on the following metrics: correctness of the answers obtained by the master and the cost incurred by the master. On the range of parameters tested, our simulations show that, even for a large number of tasks, our mechanism outperforms [15] in correctness at a similar cost. Moreover, our mechanism outperforms [24] in terms of the cost while providing similar correctness guarantees in a weaker model. We note that the repeated application of a one-shot verification mechanism was shown to be worse than [15] in [24]. Consequently, our mechanism outperforms such an approach and we omit including it in simulations for clarity.

2 Additional Related Work

We discuss the various MW models that have been studied in the literature.

MW with Malicious Workers. Distributed computations in presence of malice have been well studied (e.g. [20,34,35,44]). In [44], the workers are assumed to be malicious with some probability. To counter their effect, tasks with known outcomes are executed to detect malicious workers, which is argued to work better than voting techniques. However, later experimental work [34] showed that it takes a long time to achieve low error rates in practice. In [20], the model considers *malicious* workers who may deliberately return an incorrect result in an effort to harm the master. Workers have a predefined behavior: either they are malicious or honest. Majority voting is used to cope with malice in one-shot computations. This work was later extended [35] to many tasks. A distributed verification mechanism was introduced in [36], however, the model limits malice to 20% of the workers. None of these models study selfish (rational) behavior.

MW Model with Rational Workers. Distributed computations when workers are selfish (rational in the game-theoretic sense) have also been studied (e.g. [13,14,21–23]). In [47], all workers are assumed to be rational. Auditing and majority voting measures are used. If a non-audited computation does not yield an absolute majority, the task is recomputed to achieve reliability. Several works study coexistence of rational and malicious players [1,2,6–8,13,14,18,19,22,25,40]. For example, protocols in [19] tolerate up to k rational players that deviate from a NE, follow-up work [1] tolerates up to k colluding rational players and t Byzantine ones. The BAR model (Byzantine, Altruistic, and Rational) was introduced in [2] and later used in [37,38].

The above work applies to one-shot interactions between master and workers. That is, the system design is oriented to the reliable computation of one task.

Repeated Games and MW. In the repeated-games MW framework, the master sequentially assigns tasks to the same pool of workers. Christoforou et al. [15] (and follow-up [39]) use this framework to design a mechanism based on reinforcement-learning. In their approach, master and workers adjust their strategies for the next interaction according to the outcome of the previous one. The mechanism is shown to eventually converge to a state where the master always obtains the correct results with minimal verification. Depending on system parameters such as rewards, penalties and the workers' aspiration for profit, the time required for convergence may be long.

Our work builds upon the MW model of Fernández Anta et al. [24] and differs from it in two ways.

First, the MW model in [24] is based on infinitely repeated games [41], while we design a finitely-repeated MW game. The infinitely-repeated games approach is shown to improve upon repeated application of one-shot and reinforcement-learning mechanisms in terms of cost and reliability. However, the correctness of the model relies heavily on the assumption that the workers are uncertain about when the game ends. In particular, if the workers in [24] were to know when the interaction ends, they would stop following the equilibrium (and computing tasks) and default to cheating. Letting workers know the (possibly extended) length of the interaction a priori makes the model more realistic.

Second, the model in [24] assumes all players are perfectly rational, while we allow the presence of deviators. The term "deviators" is used in [24] to analyze deviations from the equilibrium strategy, but these deviations are never beneficial for the workers. Thus, since all workers are perfectly rational, no worker will ever deviate. In this paper, we define deviators as players who may not follow the prescribed equilibrium despite being aware of the consequence to their utility, perhaps because they do not trust the mechanism or the threats of punishment. To ensure that the punishment by followers is not just an empty threat, we incentivize followers to levy the punishment using terminal payoffs. In contrast, the punishment by peers in [24] is effectively an empty threat because imposing the punishment causes a utility loss to players, and they incur such a loss indefinitely in the infinitely repeated game.

Repeated Games and Crowdsourcing. More recently, a repeated-games approach to collect crowd sensor data was presented in [33]. Their mechanism uses a worker-reputation system, reinforcement learning to update strategies, and master verification in the presence of both rational and irrational workers. Their game model is finite but fails to address the deviation from equilibrium of rational workers when the sequence of task assignments is reaching the end.

Several crowdsourcing models focus on collusion reporting or cheating detection when there are only two workers [17,46], or considering master-worker interaction as a two-player game [31]. Such approaches are often case-by-case and do not model repeatedly outsourcing many computational tasks.

3 Preliminaries

We say that a stochastic event holds *with high probability* (*w.h.p.*), if it holds with probability at least $1 - 1/n^c$, for some constant $c > 0$, and that it holds *asymptotically almost surely* (or *a.a.s.*, for short), if it holds with probability at least $1 - 1/f(n)$, where $f(n) \in \omega(1)$.

Master-Worker Model. We consider a computing platform with a set N of processing entities called ***workers***, where $|N| = n$, and a coordinator entity called ***master***. We call this platform a master-work (MW) system.

The master assigns computing tasks to workers sequentially in ***rounds***. We assume that the master has an unbounded source of tasks that need to be computed. We consider computation of tasks that have a unique correct answer. In each round, the master assigns the same computational task to all workers. Upon receiving the results, the master decides stochastically whether to ***verify*** them (e.g. checking the solutions received or computing the result itself if needed), or accept the majority. The ***probability of verification*** by the master is denoted as p_v. In order to avoid ties, we let n be odd. We assume that the cost of verification is negligible with respect to computing the task, e.g. NP-hard problems. If the master verifies the results of the computation, it pays a ***reward*** ρ to the workers that returned the correct result, and charges a ***fine*** ϕ to those that returned and incorrect result. If the master does not verify, workers in the majority receive the reward ρ and no worker is fined.

The ***expected cost of the master*** is defined as the sum of the expected verification cost and expected payments, minus the expected fines received (since they reduce the cost).

As a design decision, our mechanism specifies a ***reward threshold*** γ on the number of workers that will be rewarded. That is, if the number of workers to be rewarded exceeds the reward threshold, the master does not reward any worker (regardless of verification). This reward threshold reflects the natural assumption that the master has a limited budget. Because receiving fines from the workers is not the goal of the system, this parameter ensures that the cost reduction due to fines does not increase the budget of the master for rewards. Note the prescribed equilibrium will ensure that the number of workers following it (the followers) stay below the threshold γ. In other words, in absence of deviations, the workers strategic decision stochastically guarantees that the number of correct answers will be at most γ.

Computing a task involves a ***cost*** for the workers, which is assumed to be the same value c for all. For each round of task assignments, the strategy of the workers is defined by their probabilistic choice to either compute the result of the task or to return a default incorrect answer to avoid the cost of computing, albeit risking to be fined. For each worker $i \in N$, let the ***probability of computing*** the task be denoted as p_{ζ_i}, and let $p_{\not\zeta_i} = 1 - p_{\zeta_i}$ be the probability of not computing.

The above protocol is followed for each round of task assignments. We summarize the notation in Table 1.

Table 1. System parameters

ρ	: reward	γ	: reward threshold
ϕ	: fine	T	: number of rounds
c	: worker computing cost	N	: set of workers ($\|N\| = n$)
p_{ζ_i}	: worker i probability of computing	n_f	: number of followers
p_v	: master probability of verifying	n_d	: number of deviators ($n_d = n - n_f$)

Repeated Game Model. We model the interactions in the MW system as a finitely repeated game with mixed strategies and terminal payoffs [27]. That is, the master assigns T computing tasks sequentially, each to the same set of n workers. The strategic choice for each worker $i \in N$ is the value of the probability of computing p_{ζ_i}.

In each round of task assignments, the master and workers proceed as in the MW model above. The master computes an equilibrium where the workers expected utility is maximized and sends the equilibrium parameters and its computation to the workers. We assume that among n workers, there are n_f ***followers*** and $n_d = n - n_f$ ***deviators***. The followers are rational in the game-theoretical sense and choose their strategies according to the prescribed equilibrium. The deviators, on the other hand, may not trust the mechanism and may choose a different strategy. The deviators motivation to steer away from equilibrium is to go unnoticed while reducing costs or increasing utilities. That is, they are not Byzantine malicious.

Upon detecting deviations, the followers change to a minmax strategy (defined below) that yields the lowest payoff that followers can force upon a deviator. The purpose of changing to a minmax strategy is to penalize the deviators for not following the equilibrium. This change of strategy lasts for a period of P rounds, which is called a ***peer-punishment*** period. P is large enough to counter any extra utility that may be obtained by the deviators. Followers compute the value P based on the detected deviation.

Without loss of generality, we assume that deviations are detected on or before the $(T - P)^{th}$ round, and that only one period of peer-punishment is needed throughout the whole computation. Otherwise, should a deviation occur after some round T_{late} such that $T_{late}+P > T$, or more than one peer-punishment period be needed, the master could assign more tasks[3] to extend the interaction period to $T_{late}+P$, or to make T much larger than the sum of all punishment periods needed respectively. Intuitively, one would expect that deviators "learn the lesson" after being punished—they confirm that deviating is indeed not worthwhile as any utility gain from deviating is nullified by the punishments. Also, we limit the number of deviations as an arbitrary number of deviations indicates malicious intent rather than bounded rationality. Furthermore, the deviators do not know whether subsequent deviations will cause punishments, and a single punishment is sufficient to establish the credibility of such a threat.

[3] Recall the assumption that the master has access to an unbounded number of computational tasks.

After the T rounds of task assignments are completed, the master pays a ***terminal payoff*** to followers to compensate them for the utility loss during the peer-punishment period, and the interaction between the master and the workers terminates.

Even though the master-worker interaction lasts a finite number of rounds, the above framework allows us to model the worker behavior as in an infinitely repeated game. That is, workers choose strategies taking into account long-term interaction. Indeed, the player behavior in infinitely repeated games is well-motivated even for finitely repeated games except in the last few rounds [41], and by introducing terminal payoffs (and extending the interaction beyond T rounds if needed) the long-term choice of strategies is not affected.

We now define the game formally. Let $G_1\langle N, (A_i), (\succsim_i)\rangle$ be a normal-form (one-round) game with set of players N, where $|N| = n$. For each player $i \in N$, $A_i = \{\zeta, \not\zeta\}$ is the action space (compute or not compute) of player $i \in N$, and $\succsim_i$ is the preference relation on the space of all workers actions $A = \times_{i\in N} A_i = \{\zeta, \not\zeta\}^N$. The preference relation of player i is represented by a ***utility function*** u_i , in the sense that $u_i(a) \geq u_i(b)$ whenever $a \succsim_i b$, for $a, b \in A$. The utility function u_i is based on the rewards and fines scheme defined in the MW model. That is, for $\#\zeta = \sum_{j\in N: A_j=\zeta} 1$ and $\#\not\zeta = \sum_{j\in N: A_j=\not\zeta} 1$,

$$u_i = \begin{cases} -c, & \text{if } A_i = \zeta \text{ and } \#\zeta > \gamma, \\ \rho - c, & \text{if } A_i = \zeta \text{ and } n/2 < \#\zeta \leq \gamma, \\ p_v\rho - c, & \text{if } A_i = \zeta \text{ and } \#\zeta < n/2, \\ -p_v\phi, & \text{if } A_i = \not\zeta \text{ and } \#\not\zeta > \gamma, \\ (1-p_v)\rho - p_v\phi, & \text{if } A_i = \not\zeta \text{ and } n/2 < \#\not\zeta \leq \gamma, \\ -p_v\phi, & \text{if } A_i = \not\zeta \text{ and } \#\not\zeta < n/2. \end{cases} \tag{1}$$

The expected utility of the workers in the normal-form game depends on the random choices of the master and other players. Following [41], we define the extended game $G_2\langle N, (S_i), (u_i)\rangle$ that differs from G_1 in that S_i is the mixed strategies space of player $i \in N$. Let $u_i : S \to \mathbb{R}$ be the ***expected utility function*** of player i, where the expected value represents player i's preferences over the space $S = \times_{i\in N} S_i$.

The overall expected utility is based on the expected utility in each of the T rounds and the terminal payoffs. We define the repeated game with terminal payoffs implemented by our mechanism as $G(T, \mathcal{W}, \omega)\langle N, (S_i), (u_i)\rangle$. We redefine the expected utility function $u_i : S^T \to \mathbb{R}$, that is, a function whose expected value represents player i's preferences over the space S^T. $\mathcal{W} \subseteq \mathbb{R}^N$ is the terminal payoffs set, and $\omega : H_T \to \mathcal{W}$ is the terminal payoffs function applied to H_T, where H_t is the set of t-tuples of elements of S, for $t \in [1, T]$. That is, H_T is the history of mixed-strategy actions at round T. We refer to the model as a strategic game for simplicity.

The ***minmax strategy*** is a follower mixed strategy (i.e., a choice of probability of computing) such that the payoff of the deviators is minimized, regardless of their strategy. The ***minmax payoff*** v_i is the lowest expected payoff of deviator i that the followers can force upon i. It is well-known (refer to Proposition 144.3 in [23]) that, for any enforceable payoff profile, that is, any payoff profile

Algorithm 1: Master algorithm with paramters $\gamma = \left\lceil n/2 + 2\sqrt{n \ln \ln n} \right\rceil$, $q = \frac{1}{2^n}\left(\sum_{j=n-\gamma}^{\gamma-1} \binom{n-1}{j} + (1-p_v)\binom{n-1}{(n-1)/2}\right)$, and $p_v = c/\phi$. Messages are sent to (and received from) all workers.

```
1  send p_ζ, q, and certificate
2  for T times do
3      send a computational task
4      upon receiving all answers do
5          verified ← true, with probability p_v or false, with probability 1 − p_v
6          if verified then
7              verify the answers
8              fine the incorrect workers
9              if correct count ≤ γ then
10                 reward workers that were correct
11         else
12             accept the answer of the majority
13             if majority count ≤ γ then
14                 reward workers in the majority
15         send list ⟨answer, count⟩ and verified
16 verify complete-information certificate received
17 send payoffs according to ω(H_T)
```

where the expected utility for each worker is at least the minmax payoff, there exists a Nash equilibrium payoff profile that all workers will follow due to long-term rationality. We assume that followers aim for ***Pareto efficiency*** (refer to Sect. 1.7 in [41]), that is, an equilibrium mixed strategy corresponding to an enforceable payoff profile that maximizes the workers expected utility.

4 Algorithmic Mechanism

In this section we describe the mechanism that implements the MW computing platform. The algorithm for the master (refer to Algorithm 1) follows the model defined in Sect. 3 in Lines 2 to 14. After computing the equilibrium p_ζ based on the parameters defined for the system (reward, fine, etc.), in Line 1 the master sends the value of p_ζ to workers together with a certificate showing the calculation of such equilibrium, and a parameter q that is a function of the number of workers n, the cost c, and the fine ϕ. In Line 15, the master sends a list of the answers received and the number of each, which the workers will use to detect deviations. Based on the complete-information certificate received from workers (i.e. H_T, rounds of punishment, correct answers, etc.), the master computes the terminal payoffs and sends to followers in Lines 16 and 17.

Algorithm 2: Algorithm for each follower i. Messages are sent to (and received from) the master.

receive p_ζ, q, and verify certificate
$sum_\delta \leftarrow 0$, $sum_u \leftarrow 0$, $r \leftarrow 0$, $t \leftarrow 0$, $p \leftarrow p_\zeta$
while $t \leq T$ **do**
  $t++$
  // computation phase
  receive computing task
  action $\leftarrow \zeta$, with probability p or $\not\zeta$, with probability $1-p$
  if action $= \zeta$ **then** result $\leftarrow$ computed task **else** result $\leftarrow$ fabricated bogus result
  send result
  // deviation-detection phase
  receive list$\langle$answer, count$\rangle$ and verified
  for *each* (answer, count) *in* list **do**
    if action $= \not\zeta$ **then** verify answer
    if answer *is correct and* count $\neq np_\zeta$ **then**
      $sum_\delta \leftarrow sum_\delta + (np_\zeta - \text{count})^2$
      $sum_u \leftarrow sum_u + u_{deviator}(\text{count}, \text{verified})$
      $r++$
  // peer-punisment phase
  if $sum_\delta \geq np_\zeta(1.6r + \ln n)$ **then** // Lemma 2
    $P \leftarrow (sum_u - \rho q + c)/(\rho q)$ // Lemma 4
    $sum_\delta \leftarrow 0$, $sum_u \leftarrow 0$, $r \leftarrow 0$, $p \leftarrow 1$
    execute computation phase P times
    $p \leftarrow p_\zeta$, $t \leftarrow t + P$
send complete-information certificate
receive terminal payoff

5 Analysis

In our analysis, we relate the system parameters to the utility of workers and the correctness of the master. First, we show bounds on the mixed strategy equilibrium, that is, a probability of computing, in which workers maximize their expected utility (refer to Lemma 1). Then, we proceed to analyze our mechanism to handle deviations. Specifically, we prove that the deviation detection method is correct w.h.p. (refer to Lemma 2), and we provide bounds on the length of the peer-punishment period and on the terminal payoffs that compensate followers for their loss of profit during such period (refer to Lemma 4). In our final theorem, we connect all the aspects of our mechanism showing additionally that the master achieves correctness in expectation and a.a.s. under some conditions.

Due to space restrictions, we defer some proofs to the full version of this paper.

To simplify exposition, we define the following probability functions and use them throughout the analysis.

$$p'_1 = \sum_{j=\frac{n+1}{2}}^{\gamma} \frac{j}{n-j} \binom{n-1}{j} p^{j-1}(1-p)^{n-j-1}\,(j-np)\,,$$

$$p'_2 = \sum_{j=1}^{\frac{n-1}{2}} \frac{j}{n-j} \binom{n-1}{j} p^{j-1}(1-p)^{n-j-1}\,(j-np)\,, \text{ and}$$

$$p'_3 = \sum_{j=n-\gamma}^{\frac{n-1}{2}} \binom{n-1}{j} p^{j-1}(1-p)^{n-j-1}\,(j-np)\,.$$

5.1 A Pareto-Efficient Repeated Game Equilibrium

Recall that, for any enforceable payoff profile, that is, any payoff profile where the expected utility for each worker is at least the minmax payoff, there exists a Nash equilibrium payoff profile that all workers will follow due to long-term rationality (refer to Proposition 144.3 in [23]). To ensure Pareto efficiency, we identify a strategy profile that maximizes the workers expected utility (refer to Sect. 1.7 in [41]).

Lemma 1. *Consider any MW system with $n > 1$ workers where $n_f = n$, $n_d = 0$, $p_v = c/\phi$, $\gamma = \left\lceil n/2 + 2\sqrt{n \ln \ln n} \right\rceil$ such that $\gamma < n$, and $\phi \geq c(|p'_2| - |p'_3|)/(p'_1 - |p'_3|)$ for any $0 \leq p \leq 1/2$. Then, the following holds.*

- *The mixed strategy equilibrium of G is such that all workers use the same probability of computing p_ζ, and such probability is in the range $1/2 < p_\zeta \leq \gamma/n$.*
- *The maximum expected utility that any worker $i \in N$ can obtain with a mixed strategy equilibrium of G is $E(u_i) = \rho q - c$, where q is in the range*

$$\frac{1}{2^n}\left(\sum_{j=n-\gamma}^{\gamma-1} \binom{n-1}{j} + (1-p_v)\binom{n-1}{(n-1)/2}\right) \leq q \leq 1.$$

5.2 Deviation-Detection Method

In this section, we establish a method for the followers to detect deviations from the prescribed equilibrium, and if so move to a peer-punishment phase. Using Chernoff bounds, the following lemma bounds the number of correct answers that should be obtained w.h.p. when workers follow an equilibrium. Based on those bounds, we provide a test that keeps track of such deviations over possibly many rounds. Thus, either if strong deviations occur over a few rounds, or slight deviations occur over many rounds, or both, the deviation is detected w.h.p.

Lemma 2. *For any MW system where all workers use the same p_ζ such that $p_\zeta > 1/2$, let X_t be the number of correct answers in round t. For any sequence of rounds (not necessarily contiguous) of task assignments $t_1, \ldots, t_r$, such that $X_t \neq np_\zeta$ for all $t \in \{t_1, \ldots, t_r\}$. For any constant $\xi > 0$, the following holds with probability at least $1 - 1/n^\xi$: $\sum_{t=t_1}^{t_r} (X_t - np_\zeta)^2 < np_\zeta(1.6r + \xi \ln n)$.*

Lemma 2 gives a method of detection for the followers. Namely, for any sequence of r rounds, possibly not contiguous, where the number of correct answers is not np_ζ, keep track of what is the difference with respect to np_ζ. If $p_\zeta < 1$, some workers do not compute the task, but they need to know how many correct answers were received to decide on the peer punishment. They can do so by verifying which, if any, of the answers sent by the master are correct, and use the corresponding count to compute the total number of correct answers. Then, whenever the inequality proved in Lemma 2 is not true, followers move to a peer-punishment phase.

5.3 Deviation Punishment and Terminal Payoffs

In this section we analyze conditions on the system parameters that enable deviation peer-punishment. That is, what is the minmax strategy that followers should use, and what is the duration of the peer-punishment phase. Recall that the minmax strategy is a follower mixed strategy, i.e. a choice of probability of computing, such that the payoff of deviators is minimized, independently of what the deviators do, for each round of peer-punishment. We define v_i as the lowest expected payoff that can be forced upon worker i by other workers. That is, for mixed strategies σ_i and σ_{-i} of worker i and workers in $N \setminus \{i\}$ respectively, we have $v_i = \min_{\sigma_{-i}} \max_{\sigma_i} E(u_i)$.

Lemma 3. *For any MW system with $n_f = n - 1$ followers and a deviator i, if all followers use $p_\zeta = 1$, it is $v_i = \max(-c, -p_v\phi)$.*

Lemma 3 states that the minmax strategy for MW systems where $n_f = n-1$ is $p_\zeta = 1$. Indeed, as long as the number of followers exceeds the reward threshold, if followers use the same minmax strategy, deviators receive the same minmax payoff. We establish this observation in Corollary 1.

Corollary 1. *For any MW system with $n_f > \gamma$ followers and $n_d = n - n_f$ deviators, if all followers follow the strategy $p_\zeta = 1$, then the minmax payoff $v_i = \max(-c, -p_v\phi)$.*

Lemma 4 relates the length of the peer-punishment phase to the extra payoff attained by a deviator until it is detected, and the utility loss from being punished.

Lemma 4. *For any MW system with $n_f > \gamma$ followers and $n_d = n - n_f$ deviators, $p_v = c/\phi$, $\gamma = \left\lceil n/2 + 2\sqrt{n \ln \ln n} \right\rceil$ such that $\gamma < n$, and $\phi \geq$*

$c(|p'_2| - |p'_3|)/(p'_1 - |p'_3|)$, *or any* $0 \leq p \leq 1/2$. *If deviations from the equilibrium are detected in rounds* $t_1, \ldots, t_r$, *then the following holds.*

(i) If followers use the minmax strategy for $P \geq \frac{\sum_{t=t_1}^{t_r} u_j(t) - \rho q + c}{\rho q - c + \min(c,\ p_v\phi)}$, *rounds of peer-punishment, where* $q = \frac{1}{2^n}\left(\sum_{j=n-\gamma}^{\gamma-1}\binom{n-1}{j} + (1-p_v)\binom{n-1}{(n-1)/2}\right)$, *and* $u_j(t)$ *is the utility obtained by any deviator* j *in each round* t *in* $t_1, \ldots, t_r$, *then any utility gain by deviators during the* $t_1, \ldots, t_r$ *rounds of deviation from the prescribed equilibrium, is lost in those* P *rounds of peer-punishment.*

(ii) To compensate the followers for their loss while using the minmax strategy, it is enough for the master to pay terminal payoffs of $\omega \geq P(\rho - c + \min(c, p_v\phi))$.

5.4 Mechanism Properties

We now establish our main result in the following theorem.

Theorem 1. *Consider the mechanism specified in Algorithms 1 and 2 under the models of Sect. 3, where the reward threshold is* $\gamma = \left\lceil n/2 + 2\sqrt{n \ln \ln n} \right\rceil$ *such that* $\gamma < n$, *the set of workers has at least* $n_f > \gamma$ *followers, the probability of verification is* $p_v = c/\phi$, *and the fine is* $\phi \geq c(|p'_2| - |p'_3|)/(p'_1 - |p'_3|)$, *for any* $0 \leq p \leq 1/2$. *Then, the following holds.*

1. *The mixed-strategy equilibrium is some probability* p_ζ *where* $1/2 < p_\zeta \leq \gamma/n$.
2. *Deviators are detected w.h.p.*
3. *The expected utility of each follower for* T *rounds is at least*

$$T\left(\frac{\rho}{2^n}\left(\sum_{j=n-\gamma}^{\gamma-1}\binom{n-1}{j} + (1-p_v)\binom{n-1}{(n-1)/2}\right) - c\right).$$

4. *The expected cost of the master is at most* $T\,(np_\zeta\rho - p_v(1-p_\zeta)\phi)$.
5. *In each round of peer-punishment, the master obtains the correct answer, and in each round of task assignments without deviations or peer-punishment,*
   (a) in expectation the master obtains the correct answer,
   (b) if $p_\zeta \geq 1/2 + \sqrt{n \ln \ln n}$ *the master obtains the correct answer a.a.s., and*
   (c) if $p_\zeta \leq 1/2 + \sqrt{n \ln \ln n}$ *the master pays less than* $\gamma\rho$ *in rewards a.a.s.*

Proof. Claims 1 to 3 follow from Lemmas 1 and 2, and the correctness of the mechanism shown in Lemma 4.

For Claim 4, given that during punishment periods followers use $p_\zeta = 1$ and $n_f > \gamma$, no worker is rewarded, the worst case cost is incurred when there is no deviation. Thus, the claimed upper bound on the expected cost is a straightforward application of the payment model specified in Sect. 3 for T rounds of task assignments without deviations.

We prove Claim 5 as follows. From Corollary 1, we know that during rounds of peer punishment followers use $p_\zeta = 1$. Given that $n_f > \gamma > n/2$, the master

obtains the correct answer during those rounds. For any round without deviations or peer-punishment, let the random variable X_ζ be the number of workers that compute. Then, the following holds.

a) It is $E(X_\zeta) = np_\zeta$. Hence, given that $p_\zeta > 1/2$ as shown in Lemma 1, it follows that the master obtains the correct answer in expectation.
b) Let $\delta = \sqrt{n \ln \ln n}/E(X_\zeta)$. Then,

$$1 - \delta = 1 - \frac{\sqrt{n \ln \ln n}}{np_\zeta} = 1 - \frac{\sqrt{n \ln \ln n}}{n/2 + \sqrt{n \ln \ln n}} = \frac{n}{2E(X_\zeta)}.$$

Given that $0 < \delta < 1$, by Chernoff bound:

$$\begin{aligned}\Pr(X_\zeta \leq n/2) &= \Pr(X_\zeta \leq E(X_\zeta)(1-\delta)) \\ &\leq \exp(-E(X_\zeta)\delta^2/2) = \exp\left(-\frac{\ln \ln n}{2p_\zeta}\right) \\ &= \left(\frac{1}{\ln n}\right)^{\frac{1}{2p_\zeta}} \leq \frac{1}{\sqrt{\ln n}}.\end{aligned}$$

Thus, $\Pr(X_\zeta \leq (n-1)/2) \leq 1/\sqrt{\ln n}$. Therefore, the master obtains the correct answer a.a.s.
c) Let $\delta = \gamma/E(X_\zeta) - 1$. Given that $0 < \delta < 1$, by Chernoff bound $\Pr(X_\zeta \geq \gamma) \leq \exp(-E(X_\zeta)\delta^2/3)$ and

$$\begin{aligned}\exp(-E(X_\zeta)\delta^2/3) &= \exp\left(-\frac{\ln \ln n}{3p_\zeta}\right) \\ &= \left(\frac{1}{\ln n}\right)^{\frac{1}{3p_\zeta}} \leq \frac{1}{\sqrt[3]{\ln n}}.\end{aligned}$$

Thus, $\Pr(X_\zeta \geq \gamma) \leq 1/\sqrt[3]{\ln n}$. Therefore, the master pays less than $\gamma\rho$ in rewards a.a.s.

6 Simulations

In this section, we present our simulations of the master and worker algorithms (Algorithms 1 and 2) in the presence of deviators. To compare the performance of our approach, we also simulated the reinforcement-learning-based mechanism in [15], and the mechanism based on infinitely-repeated games in [24].

Throughout this section, we refer to our finitely-repeated game with terminal payoffs mechanism as FRG. We refer to the *evolutionary dynamics* based approach of the mechanism in [15] as ED, and to the infinitely-repeated games approach in [24] as IRG.

Simulation Design. To the best of our knowledge, the impact of deviators on the gaurantees of ED and IRG has not been theoretically analyzed. In ED, workers update their probability of computing p_ζ in each round based on the previous

round's p_ζ, payment received, and some measure of their profit aspiration. It is assumed that all workers comply with the mechanism. On the other hand, in IRG it is assumed that, all rational workers will follow the equilibrium because of the threat of punishment.

In our experimental evaluation, we assume that 40% of workers are deviators in all three mechanisms. In FRG, the deviators and followers proceed as described in our model. In ED, the deviators and followers start with a different p_ζ, and they converge over time. In IRG, we follow the simulations in [24] and assume that the deviation is readily detected after one round (optimal detection), and that the deviators become followers after one round of punishment (optimal effect of peer punishment). For consistency we assume that deviations occur only once in all three mechanisms.

In the IRG mechanism, the expected utility of the workers is computed at equilibrium of the infinitely-repeated game. Hence, for this comparison, we assume that workers in the IRG mechanism do not know that the interaction is finite (once the number of task assignments desired by the master is completed, the interaction stops). Note that if the workers in the IRG mechanism do know the length of interaction, its correctness collapses. Comparing our approach against a stronger but less realistic execution of IRG [24] only strengthens the results of our evaluation.

We provide preliminary simulations based on the following system parameters. We evaluate all mechanisms for $n = 9$, 99, and 999 workers and the number of rounds of task assignments in the range $T \in [20, 1000]$. The payoffs scheme is evaluated for $\rho = 10$, $\phi = 10$, and $c = 2$. The master's probability of verification is set to $p_v = c/\phi = 0.2$.

For FRG, we approximate the probability of computing of the followers at equilibrium as $p_\zeta = 0.55$. For the deviators, we use $p_\zeta = 0.9$, to model their motivation to compute to avoid being fined by the master.

For ED, we set the worker aspiration for profit $a = 0.1$, learning rate $\alpha = 0.01$, the master tolerance to error $\tau = 0.5$. We set initial probabilities $p_\zeta = 0.5$ for the followers and $p_\zeta = 0.9$ for deviators, and the probability of verification $p_v = 0.2$.

We configured the parameters of IRG as follows (using the notation in [24]). The reward $WB_A = \rho = 10$, the fine $WP_C = \phi = 10$, and the cost of computing $WC_T = c = 2$. The additional master parameters in [24] are set as follows. Cost of verification MC_V, profit from being correct MB_R and cost of being wrong MP_W are all set to 0; whereas the cost of accepting an answer is $MC_A = \rho = 10$. In [24], the followers use $p_\zeta = 0.9$ and the deviators $p_\zeta \leq 0.5$. We simulate the same $p_\zeta = 0.9$ for followers; for the deviators, we set $p_\zeta = 0.5$ which is the most favorable choice for IRG.

These parameters choices satisfy the analysis of our work and [24]. A broader range of simulation parameters is left to the full version.

The simulation code was written in Python 3.6 and executed on a PC. The results presented are the average of 10 executions of each mechanism.

Discussion of Results. The results of our simulations can be seen in Figs. 1, 2, and 3. In Fig. 1, we compare the mechanisms with respect to the ratio of the

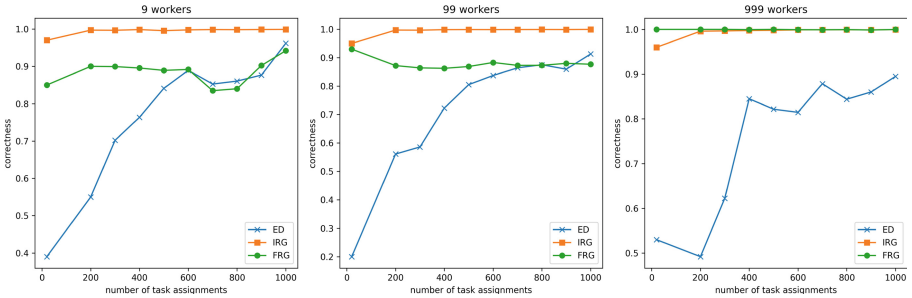

Fig. 1. Comparison of FRG, ED and IRG: number of correct answers obtained divided by the total number of task assignments vs. number of task assignments.

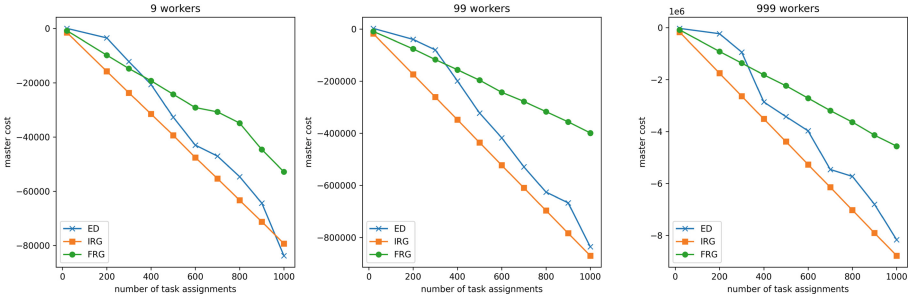

Fig. 2. Comparison of the cost incurred by the Master in FRG, ED and IRG.

number of tasks for which the master obtains the correct answer to the total number of task assignments. In Fig. 2, we compare the cost incurred by the master for each mechanism. In Fig. 3, we plot the utilities of each worker in our FRG mechanism and show that they obtain positive utilities from participating.

We observe in Figs. 1 and 2 that for any T up to 400 task assignments FRG performs much better than ED in correctness, at similar costs incurred by the master. In other words, FRG outperforms ED, unless the number of task assignments is very large. The number of task assignments T is a design choice in the mechanism. That is, the master may configure the platform to run for $T = 400$ task assignments, and hire a new pool of workers for each batch of computations.

Compared to IRG, the cost incurred by the master in FRG is much smaller, at a slightly reduced correctness rate. Note that the correctness guarantees achieved by IRG in our simulations is optimistic, because we assume that the players are unaware of the number of rounds. Thus, we conclude that for a wide range of parameters, FRG outperforms ED, and achieves similar correctness with much lower cost incurred by the master compared to IRG (assuming that the workers are unaware of the finite nature of the game).

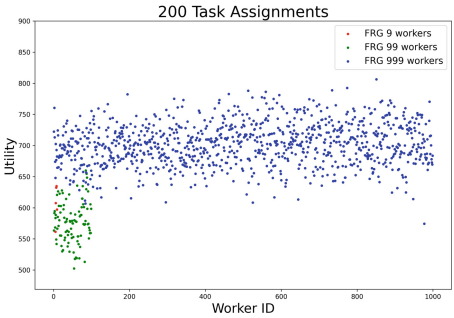

Fig. 3. Workers utility in the FRG mechanism. The x axis is the worker IDs.

Finally, in Fig. 3 we show that the expected utility of each worker is positive for each of the n values tested and for $T = 200$ (other values of T gave similar results). This confirms the feasibility of the mechanism—by participating, the workers receive non-negative utility.

7 Conclusion

In this paper, we design a finitely-repeated MW mechanism to perform verifiable crowd-computing in the presence of deviators, workers who are not perfectly rational and may not follow the prescribed equilibrium. This model better reflects real-world crowd-computing applications, where non-compliant workers exist [4, 26, 30], and contracts often have a prespecified length of interaction.

We use the notion of terminal payments to incentivize the followers (rational workers who follow the equilibrium) to punish such deviators. We prove that the master is able to obtain correct answers from such a mechanism always in expectation, and asymptotically almost surely under certain mild conditions.

Finally, we simulate our mechanism and compare it with previous approaches: ED [15] and IRG [24]. Our simulations show that our mechanism outperforms ED in terms of correctness, and IRG in terms of the cost incurred by the master.

Acknowledgement. The work presented in this manuscript was partially supported by NSF CCF 1947789, and the Pace University SR Grant and Kenan Fund.

References

1. Abraham, I., Dolev, D., Goden, R., Halpern, J.: Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In: Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing, pp. 53–62 (2006)

2. Aiyer, A.S., Alvisi, L., Clement, A., Dahlin, M., Martin, J., Porth, C.: Bar fault tolerance for cooperative services. In: Proceedings. of the 20th ACM Symposium on Operating Systems Principles, pp. 45–58 (2005)
3. Amazon.com.: Amazon Mechanical Turk. http://www.mturk.com. Accessed 2 Oct 2017
4. Anderson, D.: BOINC: a system for public-resource computing and storage. In: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, pp. 4–10 (2004)
5. Azar, P.D., Micali, S.: Super-efficient rational proofs. In: Proceedings of the 14th Annual ACM conference on Electronic Commerce (EC), pp. 29–30 (2013)
6. Babaioff, M., Feldman, M., Nisan, N.: Combinatorial agency. In: Proceedings of the 7th ACM Conference on Electronic Commerce, pp. 18–28 (2006)
7. Babaioff, M., Feldman, M., Nisan, N.: Mixed strategies in combinatorial agency. In: Proceedings of the 2nd international Workshop on Internet & Network Economics, pp. 353–364 (2006)
8. Babaioff, M., Feldman, M., Nisan, N.: Free-riding and free-labor in combinatorial agency. In: Mavronicolas, M., Papadopoulou, V.G. (eds.) SAGT 2009. LNCS, vol. 5814, pp. 109–121. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04645-2_11
9. Bielefeld, R.S.: Reexamination of the perfectness concept for equilibrium points in extensive games. In: Models of Strategic Rationality. Theory and Decision Library C, vol. 2, pp. 1–31. Springer, Dordrecht (1988). https://doi.org/10.1007/978-94-015-7774-8_1
10. Chen, J., McCauley, S., Singh, S.: Rational proofs with multiple provers. In: Proceedings of the 7th Innovations in Theoretical Computer Science Conference (ITCS), pp. 237–248 (2016)
11. Chen, J., McCauley, S., Singh, S.: Efficient rational proofs with strong utility-gap guarantees. In: Deng, X. (ed.) SAGT 2018. LNCS, vol. 11059, pp. 150–162. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99660-8_14
12. Chen, J., McCauley, S., Singh, S.: Non-cooperative rational interactive proofs. In: 27th Annual European Symposium on Algorithms (ESA 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019)
13. Christoforou, E., Fernández Anta, A., Georgiou, C., Mosteiro, M.A.: Algorithmic mechanisms for Internet supercomputing under unreliable communication. In: Proceedings of the 10th IEEE International Symposium on Network Computing and Applications, pp. 275–280 (2011)
14. Christoforou, E., Fernández Anta, A., Georgiou, C., Mosteiro, M.A.: Algorithmic mechanisms for reliable master-worker Internet-based computing. IEEE Trans. Comput. **63**(1), 179–195 (2014)
15. Christoforou, E., Fernández Anta, A., Georgiou, C., Mosteiro, M.A., Sánchez, A.: Applying the dynamics of evolution to achieve reliability in master-worker computing. Concurr. Comput. Pract. Exp. **25**(17), 2363–2380 (2013)
16. Conlisk, J.: Why bounded rationality? J. Econ. Lit. **34**(2), 669–700 (1996)
17. Dong, C., Wang, Y., Aldweesh, A., McCorry, P., van Moorsel, A.: Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 211–227 (2017)
18. Eidenbenz, R., Schmid, S.: Combinatorial agency with audits. In: Proceedings of the International Conference on Game Theory for Networks, pp. 374–383 (2009)
19. Eliaz, K.: Fault tolerant implementation. Rev. Econ. Stud. **69**, 589–610 (2002)

20. Fernández, A., Georgiou, C., Lopez, L., Santos, A.: Reliable Internet-based computing in the presence of malicious workers. Parall. Process. Lett. **22**(1) (2012)
21. Fernández Anta, A., Georgiou, C., Mosteiro, M.A.: Designing mechanisms for reliable Internet-based computing. In: Proceedings of the 7th IEEE International Symposium on Network Computing and Applications, pp. 315–324 (2008)
22. Fernández Anta, A., Georgiou, C., Mosteiro, M.A.: Algorithmic mechanisms for Internet-based master-worker computing with untrusted and selfish workers. In: Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium, pp. 1–11 (2010)
23. Fernández Anta, A., Georgiou, C., Mosteiro, M.A., Pareja, D.: Algorithmic mechanisms for reliable crowdsourcing computation under collusion. Public Lib. Sci. One **10**(3) (2015)
24. Fernández Anta, A., Georgiou, C., Mosteiro, M.A., Pareja, D.: Multi-round master-worker computing: a repeated game approach. In: Proceedings of the IEEE 35th Symposium on Reliable Distributed Systems, pp. 31–40. IEEE (2016)
25. Gairing, M.: Malicious Bayesian congestion games. In: Proceedings of the 6th Workshop on Approximation and Online Algorithms, pp. 119–132 (2008)
26. Golle, P., Mironov, I.: Uncheatable distributed computations. In: Proceedings of the Cryptographer's Track at RSA Conference 2001, pp. 425–440 (2001)
27. Gossner, O.: The folk theorem for finitely repeated games with mixed strategies. Internat. J. Game Theory **24**(1), 95–107 (1995)
28. Guo, S., Hubáček, P., Rosen, A., Vald, M.: Rational arguments: single round delegation with sublinear verification. In: Proceedings of the 5th Annual Conference on Innovations in Theoretical Computer Science (ITCS), pp. 523–540 (2014)
29. Guo, S., Hubáček, P., Rosen, A., Vald, M.: Rational sumchecks. In: Theory of Cryptography Conference, pp. 319–351 (2016)
30. Heien, E., Anderson, D., Hagihara, K.: Computing low latency batches with unreliable workers in volunteer computing environments. J. Grid Comput. **7**, 501–518 (2009)
31. Hu, Q., Wang, S., Cheng, X., Ma, L., Bie, R.: Solving the crowdsourcing dilemma using the zero-determinant strategies. IEEE Trans. Inf. Forensics Secur. **15**, 1778–1789 (2019)
32. It, F.: http://fold.it/portal/.Accessed 11 June 2016
33. Jin, X., Li, M., Sun, X., Guo, C., Liu, J.: Reputation-based multi-auditing algorithmic mechanism for reliable mobile crowdsensing. Pervasive Mob. Comput. **51**, 73–87 (2018)
34. Kondo, D., et al.: Characterizing result errors in internet desktop grids. In: Proceedings of the 13th International European Conference on Parallel and Distributed Computing, pp. 361–371 (2007)
35. Konwar, K., Rajasekaran, S., Shvartsman, A.: Robust network supercomputing with malicious processes. In: Proceedings of the 20th International Symposium on Distributed Computing, pp. 474–488 (2006)
36. Kuhn, M., Schmid, S., Wattenhofer, R.: Distributed asymmetric verification in computational grids. In: Proceedings of the 22nd IEEE International Parallel & Distributed Processing Symposium, pp. 1–10 (2008)
37. Li, H.C., et al.: Flightpath: obedience vs choice in cooperative services. In: Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation, pp. 355–368 (2008)
38. Li, H.C., et al.: Bar gossip. In: Proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation, pp. 191–204 (2006)

39. Lu, K., Yang, J., Gong, H., Li, M.: Classification-based reputation mechanism for master-worker computing system. In: Wang, L., Qiu, T., Zhao, W. (eds.) QShine 2017. LNICST, vol. 234, pp. 238–247. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78078-8_24
40. Moscibroda, T., Schmid, S., Wattenhofer, R.: When selfish meets evil: byzantine players in a virus inoculation game. In: Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing, pp. 35–44 (2006)
41. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. The MIT Press (1994)
42. Project, T.S.: http://setiathome.berkeley.edu. Accessed 11 June 2016
43. Rubinstein, A.: Modeling Bounded Rationality. MIT Press, London (1998)
44. Sarmenta, L.: Sabotage-tolerance mechanisms for volunteer computing systems. Futur. Gener. Comput. Syst. **18**(4), 561–572 (2002)
45. Treuille, A., et al.: Predicting protein structures with a multiplayer online game. Nature **466** (2010)
46. Yu, J., Li, Y.: New methods of uncheatable grid computing. Comput. Inform. **37**(6), 1293–1312 (2019)
47. Yurkewych, M., Levine, B., Rosenberg, A.: On the cost-ineffectiveness of redundancy in commercial p2p computing. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 280–288 (2005)