Internet Computing: Using Reputation to Select Workers from a Pool

Evgenia Christoforou[1,2], Antonio Fernández Anta[1], Chryssis Georgiou[3], and Miguel A. Mosteiro[4]

[1] IMDEA Networks Institute, Madrid, Spain
[2] Universidad Carlos III de Madrid, Madrid, Spain
[3] University of Cyprus, Nicosia, Cyprus
[4] Kean University, Union, NJ, USA

Abstract. The assignment and execution of tasks over the Internet is an inexpensive solution in contrast with supercomputers. We consider an Internet-based Master-Worker task computing approach, such as SETI@home. A master process sends tasks, across the Internet, to worker processors. Workers execute, and report back a result. Unfortunately, the disadvantage of this approach is the unreliable nature of the worker processes. Through different studies, workers have been categorized as either malicious (always report an incorrect result), altruistic (always report a correct result), or rational (report whatever result maximizes their benefit). We develop a reputation-based mechanism that guarantees that, eventually, the master will always be receiving the correct task result. We model the behavior of the rational workers through reinforcement learning, and we present three different reputation types to choose, for each computational round, the most reputable from a pool of workers. As workers are not always available, we enhance our reputation scheme to select the most responsive workers. We prove sufficient conditions for eventual correctness under the different reputation types. Our analysis is complemented by simulations exploring various scenarios. Our simulation results expose interesting trade-offs among the different reputation types, workers availability, and cost.

Keywords: Volunteer computing, reinforcement learning, reputation, worker reliability, task computing, worker unresponsiveness, pool of workers.

1 Introduction

Internet-based computing has emerged as an inexpensive alternative for scientific high-performance computations. The most popular form of Internet-based computing is volunteer computing, where computing resources are volunteered by the public to help solve (mainly) scientific problems. BOINC [4] is a popular platform where volunteer computing projects run, such as SETI@home [20]. Profit-seeking computation platforms, such as Amazon's Mechanical Turk [3], have also become popular. One of the main challenges for further exploiting the promise of such platforms is the untrustworthiness of the participating entities [4, 5, 16, 18].

In this work we focus on Internet-based master-worker task computing, where a master process sends tasks, across the Internet, to worker processes to compute and return the result. Workers, however, might report incorrect results. Following [9, 11], we consider three types of worker. Malicious[1] workers that always report an incorrect

[1] We call these workers malicious for compliance with Volunteer Computing [4] literature. This must not be confused with Byzantine malice assumed in classical distributed computing.

result, altruistic workers that always report a correct result, and rational workers that report a result driven by their self-interest. In addition, a worker (regardless of its type) might be unavailable (e.g., be disconnected, be busy performing other tasks, etc). Our main contribution is a computing system where the master eventually obtains always the correct task result despite the above shortcomings. Our mechanism is novel in two fronts: (*i*) it leverages the possibility of changing workers over time, given that the number of workers willing to participate is larger than the number of workers needed, and (*ii*) it is resilient to some workers being unavailable from time to time.

Worker unreliability in master-worker computing has been studied from both a classical Distributing Computing approach and a Game Theoretic one. The first treats workers as malicious or altruistic. Tasks are redundantly allocated to different workers, and voting protocols that tolerate malicious workers have been designed (e.g., [13, 19, 21]). The Game Theoretic approach views the workers as rational [1, 15, 22], who follow the strategy that would maximize their benefit. In the latter approach, incentive-based mechanisms have been developed (e.g., [14, 27]) that induce workers to act correctly.

Other works (e.g., [9, 11]) have considered the co-existence of all three types of worker. In [9], a "one-shot" interaction between master and workers was implemented. In that work, the master assigns tasks to workers without using knowledge of past interactions (e.g., on the behavior of the workers). In [11], a mechanism was designed taking advantage of the repeated interaction (rounds) of the master with the workers. The mechanism employs reinforcement learning [25] both for the master and for the workers. In each round, the master assigns a task to the same set of workers (which are assumed to be always available). The master may audit (with a cost) the responses of the workers and a reward-punishment scheme is employed. Depending on the answers, the master adjusts its probability of auditing. Rational workers cheat (i.e., respond with an incorrect result to avoid the cost of computing) with some probability, which over the rounds increases or decreases depending on the incentive received (reward or punishment). Rational workers have an aspiration level [8] which determines whether a received payoff was satisfactory or not. To cope with malicious workers (whose behavior is not affected by the above mentioned learning scheme) a reputation scheme [17] was additionally employed. The main objective is to "quickly" reach a round in the computation after which the master always receives the correct task result, with minimal auditing.

Unlike assumed in [11] (and most previous literature), in practice workers are not always available. For instance, Heien et al. [16] have found that in BOINC [4] only around 5% of the workers are available more than 80% of the time, and that half of the workers are available less than 40% of the time. In this work, we extend the work in [11] to cope with worker unavailability.

A feature that has not been leveraged in [11] and previous works is the scale of Internet-based master-worker task computing systems. For example, in BOINC [7] active workers are around a few hundred thousand. In such a large system, replicating the task and sending it to all workers is neither feasible nor practical. On the other hand, randomly selecting a small number of workers to send the task does not guarantee correctness with minimum auditing. For instance, consider a pool of workers where the malicious outnumber those needed for the computation. Then, there is a positive

probability that only malicious workers are selected and the master would have to audit always to obtain the correct result. All previous works assume the existence of a fixed/predefined set of workers that the master always contacts. In this work we consider the existence of a pool of N workers out of which the master chooses $n < N$.

Our contributions.

- We present a mechanism (in Section 3) where the master chooses the most reputable workers for each round of computation, allowing the system to eventually converge to a state where the correct result will be always obtained, with minimal auditing. Our mechanism does not require workers to be available all the time. To cope with the unavailability of the workers, we introduce a *responsiveness reputation* that conveys the percentage of task assignments to which the worker replies with an answer. The responsiveness reputation is combined with a *truthfulness reputation* that conveys the reliability of the worker. We enrich our study considering three types of truthfulness reputation. Namely, BOINC reputation (inspired in the "adaptive replication" of BOINC), EXPONENTIAL reputation (that we presented in [11]), and LINEAR reputation (inspired on the work of Sonnek et al. [24]).
- We also show formally (in Section 4) negative and positive results regarding the feasibility of achieving correctness in the long run in the absence of rational workers. Specifically, we show configurations (worker types, availability, etc.) of the pool of workers such that correctness cannot be achieved unless the master always audits, and the existence of configurations such that eventually correctness is achieved forever with minimal auditing.
- We evaluate experimentally (in Section 5) our mechanism with extensive simulations under various conditions. Our simulations complement the analysis taking into account scenarios where rational workers exist. The different reputation types are compared showing trade-offs between reliability and cost.

2 Model

Master-Worker Framework. We consider a master and a pool (set) of workers $\mathcal{N}$, where $|\mathcal{N}| = N$. The computation is broken into *rounds* $r = 1, 2, ...$ In each round r, the master selects a set W^r of $n < N$ workers, and sends them a task. The workers in W^r are supposed to compute the task and return the result, but may not do so (e.g., unavailable computing other task). The master, after waiting for a fixed time t, proceeds with the received replies. Based on those replies, the master must decide which answer to take as the correct result for this round. The master employs a reputation mechanism put in place to choose the n most reputable workers in every round. We assume that tasks have a unique solution; although such limitation reduces the scope of application of the presented mechanism [26], there are plenty of computations where the correct solution is unique: e.g., any mathematical function.

Worker unavailability. In Internet-based master-worker computations, and especially in volunteering computing, workers are not always available to participate in a computation [16] (e.g., they are off-line for a particular period of time). We assume that each worker's availability is stochastic and independent of other workers. Formally, we let $d_i > 0$ be the probability that the master receives the reply from worker i within time t (provided that the worker was chosen by the master to participate in the computation for

the given round r, i.e., $i \in W^r$). In other words, this is the probability that the worker is available to compute the task assigned.

Worker types. We consider three types of workers: *rational, altruistic,* and *malicious*. Rational workers are selfish in a game-theoretic sense and their aim is to maximize their utility (benefit). In the context of this paper, a worker is *honest* in a round, when it truthfully computes and returns the correct result, and it *cheats* when it returns some incorrect value. Altruistic and malicious workers have a predefined behavior: to always be honest and cheat respectively. Instead, a rational worker decides to be honest or cheat depending on which strategy maximizes its utility. We denote by $p_{Ci}(r)$ the probability of a rational worker i cheating in round r, provided that $i \in W^r$. The worker adjusts this probability over the course of the multiround computation using a reinforcement learning approach. The master is not aware of each worker type, neither of the distribution over types. That is, our mechanism does not rely on any statistical information.

While workers make their decision individually and with no coordination, following [13, 21], we assume that all the workers that cheat in a round return the same incorrect value. This yields a worst case scenario for the master to obtain the correct result using a voting mechanism. This assumption subsumes models where cheaters do not necessarily return the same answer, and it can be seen as a weak form of collusion.

Auditing, Payoffs, Rewards and Aspiration. When necessary, the master employs *auditing* and *reward/punish* schemes to induce the rational workers to be honest. In each round, the master may decide to audit the response of the workers, at a cost. In this work, auditing means that the master computes the task by itself, and checks which workers have been honest. We denote by $p_{\mathcal{A}}(r)$ the probability of the master auditing the responses of the workers in round r. The master can change this auditing probability over the course of the computation, but restricted to a minimum value $p_{\mathcal{A}}^{min} > 0$. When the master audits, it can accurately reward and punish workers. When the master does not audit, it rewards only those in the weighted majority (see below) of the replies received and punishes no one.

We consider three worker payoff parameters: (a) $WP_{\mathcal{C}}$: worker's punishment for being caught cheating, (b) $WC_{\mathcal{T}}$: worker's cost for computing a task, and (c) $WB_{\mathcal{Y}}$: worker's benefit (typically payment) from the master's reward. As in [8], we also assume that a worker i has an *aspiration* a_i, which is the minimum benefit that worker i expects to obtain in a round. We assume that the master has the freedom of choosing $WB_{\mathcal{Y}}$ and $WP_{\mathcal{C}}$ with the goal of satisfying *eventual correctness*, defined next. E.g., in order to motivate the worker to participate in the computation, the master ensures that $WB_{\mathcal{Y}} - WC_{\mathcal{T}} \geq a_i$; in other words, the worker has the potential of its aspiration to be covered even if it computes the task.

Eventual Correctness. The goal of the master is to eventually obtain a reliable computational platform: After some finite number of rounds, the system must guarantee that the master obtains the correct task results in every round with probability 1 and audits with probability $p_{\mathcal{A}}^{min}$. We call such property *eventual correctness*. Observe that eventual correctness implies that eventually the master receives at least one (correct) reply in every round.

Reputation. The reputation of each worker is measured and maintained by the master. Reputation is used by the master to cope with the uncertainty about the workers'

truthfulness and availability. In fact, the workers are unaware that a reputation scheme is in place, and their interaction with the master does not reveal any information about reputation; i.e., the payoffs do not depend on a worker's reputation. The master wants to assign tasks to workers that are reliable, that is, workers that are both responsive *and* truthful. Hence, we consider the worker's reputation as the product of two factors: responsiveness reputation and truthfulness reputation. Thus, the malicious workers will obtain a low reputation fast due to their low truthfulness reputation, and also the workers that are generally unavailable will get a low reputation due to their low responsiveness reputation. Consequently, these workers will stop being chosen by the master.

More formally, we define the reputation of a worker i as $\rho_i = \rho_{rs_i} \cdot \rho_{tr_i}$, where ρ_{rs_i} represents the responsiveness reputation and ρ_{tr_i} the truthfulness reputation of worker i. We also define the reputation of a set of workers $Y \subseteq W$ as the aggregated reputation of all workers in Y. That is, $\rho_Y(r) = \sum_{i \in Y} \rho_i(r)$.

In this work, we consider three truthfulness reputation types: LINEAR, EXPONENTIAL, and BOINC. In the LINEAR reputation type (introduced in [24]) the reputation changes at a linear rate. The EXPONENTIAL reputation type (introduced in [11]) is "unforgiving", in the sense that the reputation of a worker caught cheating will never increase. The reputation of a worker in this type changes at an exponential rate. The BOINC reputation type is inspired by BOINC [6]. In the BOINC system this reputation method is used to avoid redundancy if a worker is considered honest[2]. For the responsiveness reputation we use the LINEAR reputation, adjusted for responses. For the worker's availability it is natural to use a "forgiving" reputation, especially when considering volunteer computing. For the detailed description of the reputation types we introduce some necessary notation as follows.

$select_i(r)$: the number of rounds the master selected worker i up to round r.
$reply\_select_i(r)$: the number of rounds up to round r in which worker i was selected and the master received a reply from i.
$audit\_reply\_select_i(r)$: the number of rounds up to round r where the master selected worker i, received its reply and audited.
$correct\_audit_i(r)$: the number of rounds up to round r where the master selected worker i, received its reply, audited and i was truthful.
$streak_i(r)$: the number of rounds $\leq r$ in which worker i was selected, audited, and replied correctly after the latest round in which it was selected, audited, and caught cheating.

Then, the reputation types we consider are as follows.

Responsiveness reputation: $\rho_{rs_i}(r) = \frac{reply\_select_i(r)+1}{select_i(r)+1}$.
Truthfulness reputation:

LINEAR**:** $\rho_{tr_i}(r) = \dfrac{correct\_audit_i(r) + 1}{audit\_reply\_select_i(r) + 1}$.

EXPONENTIAL**:** $\rho_{tr_i}(r) = \varepsilon^{audit\_reply\_select_i(r) - correct\_audit_i(r)}$, where $\varepsilon \in (0, 1)$.

BOINC**:** $\rho_{tr}(r) = \begin{cases} 0, & \text{if } streak(r) < 10. \\ 1 - \frac{1}{streak(r)}, & \text{otherwise.} \end{cases}$

[2] In BOINC, honesty means that the worker's task result agrees with the majority, while in our work this decision is well-founded, since the master audits.

All workers are assumed to have the same initial reputation before the master interacts with them. The goal of the above definitions is for workers who are responsive *and* truthful to eventually have high reputation, whereas workers who are not responsive *or* not truthful, to eventually have low reputation.

3 Reputation-based Mechanism

We now present our reputation-based mechanism. The mechanism is composed by an algorithm run by the master and an algorithm run by each worker.

Master's Algorithm. The algorithm followed by the master, Algorithm 1, begins by choosing the initial probability of auditing and the initial reputation (same for all workers). The initial probability of auditing will be set according to the information the master has about the environment (e.g., workers' initial p_C). For example, if it has no information about the environment, a natural approach would be to initially set $p_{\mathcal{A}} = 0.5$ or $p_{\mathcal{A}} = 1$ (as a more conservative approach). The master also chooses the truthfulness reputation type to use.

At the beginning of each round, the master chooses the n most reputable workers out of the total N workers (breaking ties uniformly at random) and sends them a task T. In the first round, since workers have the same reputation, the choice is uniformly at random. Then, after waiting t time to receive the replies from the selected workers, the master proceeds with the mechanism. The master updates the responsiveness reputation and audits the answers with probability $p_{\mathcal{A}}$. In the case the answers are not audited, the master accepts the value returned by the weighed majority. In Algorithm 1, m is the value returned by the weighted majority and R_m is the subset of workers that returned m. If the master audits, it updates the truthfulness reputation and the audit probability for the next round. Then, the master rewards/penalizes the workers as follows. If the master audits and a worker i is a cheater (i.e., $i \in F$), then $\mathit{\Pi}_i = -\mathit{WP}_{\mathcal{C}}$; if i is honest, then $\mathit{\Pi}_i = \mathit{WB}_{\mathcal{Y}}$. If the master does not audit, and i returns the value of the weighted majority (i.e., $i \in R_m$), then $\mathit{\Pi}_i = \mathit{WB}_{\mathcal{Y}}$, otherwise $\mathit{\Pi}_i = 0$.

In the update of the audit probability $p_{\mathcal{A}}$, we include a threshold, denoted by τ, that represents the master's *tolerance* to cheating (typically, we will assume $\tau = 1/2$ in our simulations). If the ratio of the aggregated reputation of cheaters with respect to the total is larger than τ, $p_{\mathcal{A}}$ is increased, and decreased otherwise. The amount by which $p_{\mathcal{A}}$ changes depends on the difference between these values, modulated by a *learning rate* α_m [25]. This latter value determines to what extent the newly acquired information will override the old information. For example, if $\alpha_m = 0$ the master will never adjust $p_{\mathcal{A}}$.

Workers' Algorithm. Altruistic and malicious workers have predefined behaviors. When they are selected and receive a task T from the master, if they are available, they compute the task (altruistic) or fabricate an arbitrary solution (malicious), replying accordingly. If they are not available, they do not reply. Rational workers run the algorithm described in Algorithm 2. The execution of the algorithm begins with a rational worker i deciding an initial probability of cheating p_{Ci}. Then, the worker waits to be selected and receive a task T from the master. When so, and if it is available at the time, then with probability $1 - p_{Ci}$, worker i computes the task and replies to the master with the correct answer. Otherwise, it fabricates an answer, and sends the incorrect response

Algorithm 1 Master's Algorithm

$p_{\mathcal{A}} \leftarrow x$, *where* $x \in [p_{\mathcal{A}}^{min}, 1]$
for $i \leftarrow 0$ *to* N **do**
    $select_i \leftarrow 0$; $reply\_select_i \leftarrow 0$; $audit\_reply\_select_i \leftarrow 0$; $correct\_audit_i \leftarrow 0$; $streak_i \leftarrow 0$
    $\rho_{rs_i} \leftarrow 1$; *initialize* ρ_{tr_i} *// initially all workers have the same reputation*
for $r \leftarrow 1$ *to* ∞ **do**
    $W^r \leftarrow \{i \in \mathcal{N} : i$ *is chosen as one of the* n *workers with the highest* $\rho_i = \rho_{rs_i} \cdot \rho_{tr_i}\}$
    $\forall i \in W^r : select_i \leftarrow select_i + 1$
    send *a task* T *to all workers in* W^r
    collect replies from workers in W^r **for** t *time*
    wait for t *time collecting replies as received from workers in* W^r
    $R \leftarrow \{i \in W^r :$ *a reply from* i *was received by time* $t\}$
    $\forall i \in R : reply\_select_i \leftarrow reply\_select_i + 1$
    update *responsiveness reputation* ρ_{rs_i} *of each worker* $i \in W^r$
    audit *the received answers with probability* $p_{\mathcal{A}}$
    if *the answers were not audited* **then**
        accept *the value* m *returned by workers* $R_m \subseteq R$,
            where $\forall m', \rho_{tr_{R_m}} \geq \rho_{tr_{R_{m'}}}$ *// weighted majority of workers in* R
    else *// the master audits*
        foreach $i \in R$ **do**
            $audit\_reply\_select_i \leftarrow audit\_reply\_select_i + 1$
            if $i \in F$ **then** $streak_i \leftarrow 0$ *// $F \subseteq R$ is the set of responsive workers caught cheating*
            else $correct\_audit_i \leftarrow correct\_audit_i + 1$, $streak_i \leftarrow streak_i + 1$ *// honest responsive workers*
            update *truthfulness reputation* ρ_{tr_i} *// depending on the type used*
        if $\rho_{tr_R} = 0$ **then** $p_{\mathcal{A}} \leftarrow \min\{1, p_{\mathcal{A}} + \alpha_m\}$
        else
            $p'_{\mathcal{A}} \leftarrow p_{\mathcal{A}} + \alpha_m(\rho_{tr_F}/\rho_{tr_R} - \tau)$
            $p_{\mathcal{A}} \leftarrow \min\{1, \max\{p_{\mathcal{A}}^{min}, p'_{\mathcal{A}}\}\}$
    $\forall i \in W^r$: **return** Π_i *to worker* i *// the payoff of workers in* $W^r \setminus R$ *is zero*

Algorithm 2 Algorithm for Rational Worker i

$p_{Ci} \leftarrow y$, *where* $y \in [0, 1]$
repeat forever
    wait for *a task* T *from the master*
    if *available* **then**
        decide whether to cheat or not independently with distribution $P(cheat) = p_{Ci}$
        if *the decision was to cheat* **then**
            send *arbitrary solution to the master*
            get payoff Π_i
            $p_{Ci} \leftarrow \max\{0, \min\{1, p_{Ci} + \alpha_w(\Pi_i - a_i)\}\}$
        else
            send $compute(T)$ *to the master*
            get payoff Π_i
            $p_{Ci} \leftarrow \max\{0, \min\{1, p_{Ci} - \alpha_w(\Pi_i - WC_{\mathcal{T}} - a_i)\}\}$

to the master. After receiving its payoff, worker i changes its p_{Ci} according to payoff Π_i, the chosen strategy (cheat or not cheat), and its aspiration a_i. Similarly to the master, the workers have a *learning rate* α_w. We assume that all workers have the same learning rate, that is, they learn in the same manner (in [25], the learning rate is called step-size). In a real platform the workers learning rate can slightly vary (since workers in these platforms have similar profiles), making some worker more or less susceptible to reward and punishment. Using the same learning rate for all workers is representative of what happens in a population of different values with small variations around some mean.

4 Analysis

In this section, we prove some properties of the system. We start by observing that, in order to achieve eventual correctness, it is necessary to change workers over time. [3]

Observation 1 *If the number of malicious workers is at least n and the master assigns the task to the same workers in all rounds, eventual correctness cannot be guaranteed.*

The intuition behind this observation is that there is always a positive probability that the master will select n malicious workers at the first round and will have to remain with the same workers. This observation justifies that the master has to change its choice of workers if eventual correctness has to be guaranteed. We apply the natural approach of choosing the n workers with the largest reputation among the N workers in the pool (breaking ties randomly). In order to guarantee eventual correctness we need to add one more condition regarding the availability of the workers.

Observation 2 *To guarantee eventual correctness at least one non-malicious worker i must exist with $d_i = 1$.*

To complement the above observations, we show now that there are sets of workers with which eventual correctness is achievable using the different reputation types (LINEAR and EXPONENTIAL as truthfulness reputations) defined and the master reputation-based mechanism in Algorithm 1.

Theorem 3. *Consider a system in which workers are either altruistic or malicious and there is at least one altruistic worker i with $d_i = 1$ in the pool. Eventual correctness is satisfied if the mechanism of Algorithm 1 is used with the responsiveness reputation and any of the truthfulness reputations* LINEAR *or* EXPONENTIAL.

The intuition behind the proof is that thanks to the decremental way in which the reputation of a malicious worker is calculated at some point the altruistic worker i with full responsiveness ($d_i = 1$) will be selected and have a greater reputation than the aggregated reputation of the selected malicious workers. A similar result does not hold if truthfulness reputation of type BOINC is used. In this case, we have found that it is not enough that one altruistic worker with full availability exists, but also the number of altruistic workers with partial availability have to be considered.

Theorem 4. *Consider a system in which workers are either altruistic or malicious and there is at least one altruistic worker i with $d_i = 1$ in the pool. In this system, the mechanism of Algorithm 1 is used with the responsiveness reputation and the truthfulness reputation* BOINC. *Then, eventual correctness is satisfied if and only if the number of altruistic workers with $d_j < 1$ is smaller than n.*

[3] The omitted proofs can be found at http://arxiv.org/abs/1603.04394.

Proof. In this system, it holds that every malicious worker k has truthfulness reputation $\rho_{tr_k} = 0$ forever, since the replies that the master receives from it (if any) are always incorrect. Initially, altruistic workers also have zero truthfulness reputation. An altruistic worker j has positive truthfulness reputation after it is selected, and its reply is received and audited by the master 10 times. Observe that, once that happens, the truthfulness reputation of worker j never becomes 0 again. Also note that the reponsiveness reputation never becomes 0. Hence, the first altruistic workers that succeed in raising their truthfulness reputation above zero are always chosen in future rounds. While there are less than n workers with positive reputation, the master selects at random from the zero-reputation workers in every round. Then, eventually (in round r_0) there are n altruistic workers with positive reputation, or there are less than n but all altruistic workers are in that set. After then, no new altruistic worker increase its reputation (in fact, is ever selected), and the set of altruistic selected workers is always the same.

If the number of altruistic workers with $d_j < 1$ is smaller than n, since worker i has $d_i = 1$, after round r_0 among the selected workers there are altruistic workers with $d_j = 1$ and positive reputation. Then, in every round there is going to be a weighted majority of correct replies, and eventual correctness is guaranteed.

If, on the other hand, the number of altruistic workers with $d_j < 1$ is at least n, there is a positive probability that all the n workers with positive reputation are from this set. Since there is a positive probability that n altruistic workers with $d_j < 1$ are selected in round r_0 with probability one the worker i with $d_i = 1$ will never be selected. If this is the case, eventual correctness is not satisfied (since there is a positive probability that the master will not receive a reply in a round). Assume otherwise and consider that after round r'_0 it holds that $p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$. Then, in every round after r'_0 there is a positive probability that the master receives no reply from the selected workers and it does not audit, which implies that it does not obtain the correct result. □

This result is rather paradoxical, since it implies that a system in which all workers are altruistic (one with $d_i = 1$ and the rest with $d_j < 1$) does not guarantee eventual correctness, while a similar system in which the partially available workers are instead malicious does. This paradox comes to stress the importance of selecting the right truthfulness reputation. Theorem 4 shows a positive correlation among a truthfulness reputation with the availability factor of a worker in the case a large number of altruistic workers.

5 Simulations

Theoretical analysis is complemented with illustrative simulations on a number of different scenarios for the case of full and partial availability. The simulated cases give indications on the values of some parameters (controlled by the master, namely the type of reputation and the initial $p_{\mathcal{A}}$) under which the mechanism performs better. The rest of the parameters of the mechanism and the scenarios presented are essentially based on the observations extracted from [2, 12], and are rather similar to our earlier work [11]. We have developed our own simulation setup by implementing our mechanism (Algorithms 1 and 2, and the reputation types discussed above) using C++. The simulations were executed on a dual-core AMD Opteron 2.5GHz processor, with 2GB RAM, running CentOS version 5.3.

For simplicity, we consider that all workers have the same aspiration level $a_i = 0.1$, although we have checked that with random values the results are similar to those presented here, provided their variance is not very large ($a_i \pm 0.1$). We consider the same learning rate for the master and the workers, i.e., $\alpha = \alpha_m = \alpha_w = 0.1$. Note that the learning rate, as discussed for example in [25] (called step-size there), is generally set to a small constant value for practical reasons. We set $\tau = 0.5$ (c.f., Sect. 3; also see [10]), $p_{\mathcal{A}}^{min} = 0.01$, and $\varepsilon = 0.5$ in reputation EXPONENTIAL. We assume that the master does not punish the workers $WP_{\mathcal{C}} = 0$, since depending on the platform used this might not be feasible, and hence more generic results are considered. Also we consider that the cost of computing a task is $WC_{\mathcal{T}} = 0.1$ for all workers and, analogously, the master is rewarding the workers with $WB_{\mathcal{Y}} = 1$ when it accepts their result (for simplicity no further correlation among these two values is assumed). The initial cheating probability used by rational workers is $p_{Ci} = 0.5$ and the number of selected workers is set to $n = 5$.

The first batch of simulations consider the case when the workers are fully available (i.e, all workers have $d = 1$), and the behavior of the mechanism under different pool sizes is studied. The second batch considers the case where the workers are partially available.

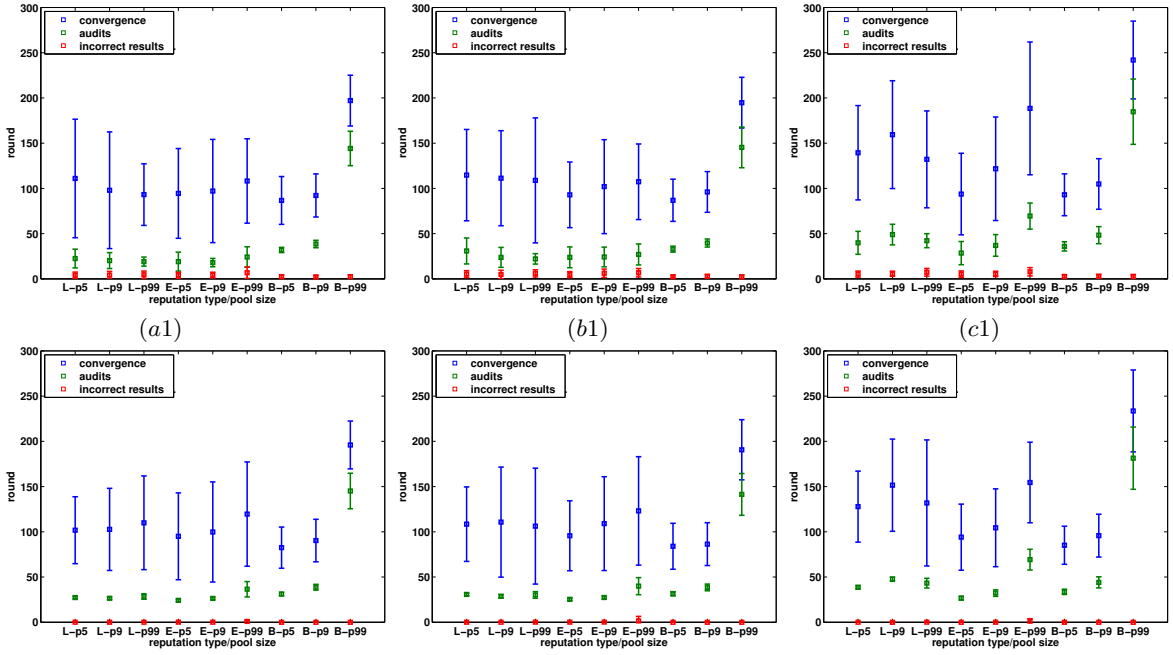
Full Availability. Assuming full worker availability we attempt to identify the impact of the pool size on different metrics: (1) the number of rounds, (2) number of auditing rounds, and (3) number of incorrect results accepted by the master, all of them measured until the system reaches convergence (the first round in which $p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$)[4]. Additionally, we are able to compare the behavior of the three truthfulness reputation types, showing different trade-off among reliability and cost.

We have tested the mechanism proposed in this paper with different initial $p_{\mathcal{A}}$ values. We present here two interesting cases of initial audit probability, $p_{\mathcal{A}} = 0.5$ and $p_{\mathcal{A}} = 1$. The first row of Figure 1 (plots (a1) to (c1)) presents the results obtained in the simulations with initial $p_{\mathcal{A}} = 0.5$ and the second row (plots (a2) to (c2)) the case $p_{\mathcal{A}} = 1$. The simulations in this section have been done for systems with only rational and malicious workers, with 3 different ratios between these worker types (ratios 5/4, 4/5, and 1/8), with different pool sizes ($N = \{5, 9, 99\}$), and for the 3 truthfulness reputation types. These ratios consider the three most "critical" cases in which malicious workers can influence the results.

A general conclusion we can extract from the first row of Figure 1 (plots (a1) to (c1)) is that, independently of the ratio between malicious and rational workers, the trend that each reputation type follows for each of the different pool size scenarios is the same. (When the ratio of rational/malicious is 1/8 this trend is more noticeable.) Reputation LINEAR does not show a correlation between the pool size and the evaluation metrics. This is somewhat surprising given that other two reputation types are impacted by the pool size.

For reputation EXPONENTIAL and BOINC we can observe that, as the pool size increases, the number of rounds until convergence also increases. It seems like, for these reputation types, many workers from the pool have to be selected and audited before convergence. Hence, with a larger pool it takes more rounds for the mechanism to select

[4] As we have seen experimentally, first the system reaches a reliable state and then $p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$.



$(a1)$ $(b1)$ $(c1)$

$(a2)$ $(b2)$ $(c2)$

Fig. 1: Simulation results with full availability. First row plots are for initial $p_{\mathcal{A}} = 0.5$. Second row plots are for initial $p_{\mathcal{A}} = 1$. The bottom (red) errorbars present the number of incorrect results accepted until convergence ($p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$), the middle (green) errorbars present the number of audits until convergence; and finally the upper (blue) errorbars present the number of rounds until convergence, in 100 instantiations. In plots (a1) and (a2) the ratio of rational/malicious is 5/4. In plots (b1) and (b2) the ratio of rational/malicious is 4/5. In plots (c1) and (c2) the ratio of rational/malicious is 1/8. The x-axes symbols are as follows, L: LINEAR, E: EXPONENTIAL and B: BOINC reputation; p5: pool size 5, p9: pool size 9 and p99: pool size 99.

and audit these workers, and hence to establish valid reputation for the workers and to reinforce the rational ones to be honest. For both reputation types (EXPONENTIAL and BOINC) this is a costly procedure also in terms of auditing for all rational/malicious ratios. (The effect on the number of audits is more acute for reputation BOINC as the pool size increases.) As for the number of incorrect results accepted until convergence, with reputation EXPONENTIAL they still increase with the pool size. However, reputation BOINC is much more robust with respect to this metric, essentially guaranteeing that no incorrect result is accepted.

Comparing now the performance of the different reputation types based on our evaluation metrics, it seems that reputation LINEAR performs better when the size of the pool is big compared to the other two reputation types. On the other hand reputation types EXPONENTIAL and BOINC perform slightly better when the pool size is small. Comparing reputation types EXPONENTIAL and BOINC, while reputation BOINC shows that has slightly faster convergence, this is traded for at least double auditing than reputation EXPONENTIAL. On the other hand, reputation EXPONENTIAL is accepting a greater number of incorrect results until convergence. This is a clear ex-

ample of the trade-off between convergence time, number of audits, and number of incorrect results accepted.

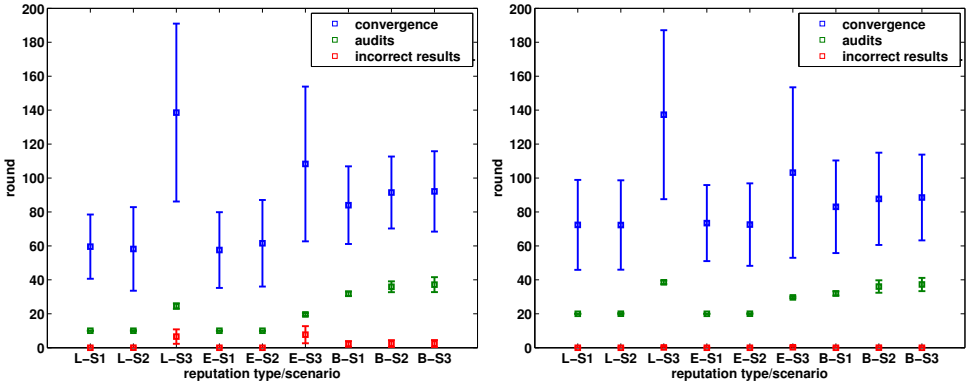
Similar conclusions can be drawn when the master decides to audit with $p_{\mathcal{A}} = 1$ initially, see Figure 1 (a2) - (c2). The only difference is that the variance, of the different instantiations on the three metrics is smaller. Hence, choosing $p_{\mathcal{A}} = 1$ initially is a "safer" strategy for the master.

Partial Availability. Assuming now partial worker availability (i.e, workers may have $d < 1$), we attempt to identify the impact of the unavailability of a worker on four different metrics: (1) the number of rounds, (2) number of auditing rounds, and (3) number of incorrect results accepted by the master, all until the system reaches convergence. In addition, we obtain (4) the number of incorrect results accepted by the master *after* the system reaches convergence (which was zero in the previous section). Moreover, we are able to identify how suitable each reputation is, under different workers' ratio and unavailability probabilities.

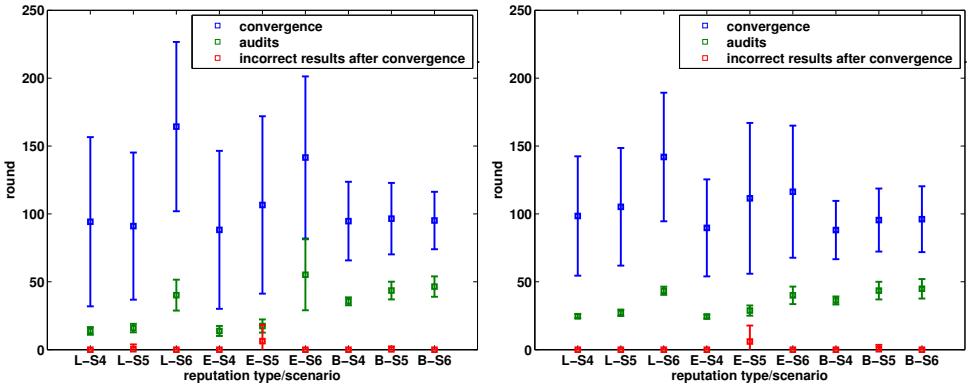
We keep the pool size fixed to $N = 9$, and the number of selected workers fixed to $n = 5$; and we analyze the behavior of the system in a number of different scenarios where the workers types and availabilities vary. The depicted scenarios present the cases of initial audit probability: $p_{\mathcal{A}} = \{0.5, 1\}$.

Figure 2 (a1)-(b1) compares a base case where all workers are altruistic with $d = 1$ (scenario S1) with scenarios where 1 altruistic worker exists with $d = 1$ and the rest of the workers are either altruistic (scenario S2) or malicious (scenario S3) with a partial availability $d = 0.5$. Our base case S1 is the optimal scenario, and the mechanism should have the best performance with respect to metrics (1)-(3); this is confirmed by the simulations as we can observe. For scenario S2, where the 8 altruistic workers have $d = 0.5$, reputations LINEAR and EXPONENTIAL are performing as good as the base case. While BOINC is performing slightly worse than the base case. Comparing the different reputation types for scenarios S1 and S2, it is clear that, for all metrics, LINEAR and EXPONENTIAL are performing better than BOINC. Moving on to scenario S3, where 8 malicious workers with $d = 0.5$ exist, as expected, the mechanism is performing worse according to our reputation metrics. What is interesting to observe, though, is that reputation BOINC is performing much better than the other two reputation types. It is surprising to observe, for reputation BOINC, how close are the results for scenario S2 and especially scenario S3 to the base case S1. We believe that this is due to the nature of reputation BOINC, which keeps reputation to zero until a reliability threshold is achieved. From the observation of Figure 2 (a1)-(b1), we can conclude that, if there is information on the existence of malicious workers in the computation, a "safer" approach would be the use of reputation BOINC. The impact of $p_{\mathcal{A}}$ on the performance of the mechanism, in the particular scenarios, as it is shown on Figure 2 (a1)-(b1), in all cases setting $p_{\mathcal{A}} = 0.5$ initially improves the performance of the mechanism.

The results of Figure 2 (a1)-(b1) are confirmed by Theorem 3. Through the simulation results, we have observed that eventual correctness happens (i.e., no more erroneous results are further accepted) when the system converges, for the depicted scenarios. As for Theorem 4 we have observed that, although the condition of having 5 altruistic with $d = 1$ is not the case for scenarios S2 and S3, in the particular scenarios simulated the system was able to reach eventual correctness. Although from the de-



$(a1)$ $(b1)$



$(a2)$ $(b2)$

Fig. 2: Simulation results with partial availability: (a1)-(a2) initial $p_{\mathcal{A}} = 0.5$, (b1)-(b2) initial $p_{\mathcal{A}} = 1$. For (a1)-(b1) The bottom (red) errorbars present the number of incorrect results accepted until convergence ($p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$). For (a2)-(b2) the bottom (red) errorbars present the number of incorrect results accepted after convergence. For all plots, the middle (green) errorbars present the number of audits until convergence; and finally the upper (blue) errorbars present the number of rounds until convergence, in 100 instantiations. The x-axes symbols are as follows, L: reputation LINEAR, E: reputation EXPONENTIAL, B: reputation BOINC, S1: 9 altruistic workers with $d = 1$, S2: 1 altruistic with $d = 1$ and 8 altruistic workers with $d = 0.5$, S3: 1 altruistic with $d = 1$ and 8 malicious workers with $d = 0.5$, S4: 9 rational workers with $d = 1$, S5: 1 rational with $d = 1$ and 8 rational workers with $d = 0.5$, S6: 1 rational with $d = 1$ and 8 malicious workers with $d = 0.5$.

picted scenarios reputation BOINC seems like is a good approach, theory tells us that it can only be used when we have info on the workers types.

Figure 2 (a2)-(b2), depicts more scenarios with different workers types ratios, in the presence of rational and malicious workers. Following the same methodology as before, we compare a base case (scenario S4) where all workers are rational with $d = 1$, with a scenarios where one rational with $d = 1$ exists and the rest are rational (scenario S5) or malicious (scenario S6) with $d = 0.5$. We can observe that in the base scenario S4, the mechanism is performing better than in the other two scenarios, for reputation metrics (1),(2) and (4), independently of the reputation type. What we observe is that

the most difficult scenario for the mechanism to handle is scenario S5, independently of the reputation type, because, although the system converges, eventual correctness has not been reached and the master is accepting incorrect replies for a few more rounds before reaching eventual correctness. This is due to the ratio of the workers' type, and some rational workers that have not been fully reinforced to a correct behavior may have a greater reputation than the rational worker with $d = 1$, while the master has already dropped $p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$. That would mean that the master would accept the result of the majority that might consist of rational workers that cheat. As we can see, EXPONENTIAL is performing worse than the other two types, based on metric (4). As for reputation LINEAR we can see that, for scenarios S4 and S5, although the variation on the convergence round is greater than reputation BOINC, this is traded for half the auditing that reputation BOINC requires. As for scenario S6 (with malicious workers), reputation LINEAR converges much slower, while the number of audits is roughly the same, compared to reputation BOINC. This observation gives a slight advantage to reputation BOINC for scenario S6, while reputation LINEAR has an advantage on S5.

Discussion. One conclusion that is derived by our simulations is that, in the case of full availability, reputation BOINC is not a desirable reputation type if the pool of workers is large. As simulations showed us, convergence is slow, and expensive in terms of auditing. One could select one of the other two reputation types (according to the available information on the ratio of workers' type), since accepting a few more incorrect results is traded for fast eventual correctness and low auditing cost. Additionally, in the scenario with full availability we have noticed that, selecting initially $p_{\mathcal{A}} = 1$ is a "safer" option to have small number of incorrect results accepted, if no information on the system is known and the master is willing to invest a bit more on auditing.

For the case of partial availability, the simulations with only altruistic or with altruistic and malicious converged in all cases. This was expected due to the analysis in all cases except in S2 with reputation BOINC, when we expected to see some rounds after convergence with no replies. The fact is that the altruistic worker with full availability was able to be selected forever in al cases. Simulations have also shown that, in the presence of malicious and altruistic workers, reputation BOINC has an advantage compared to the other two types. Finally, it is interesting to observe that, in the partial availability case with only rational workers, our mechanism has not reached eventual correctness when the system has converged, but a few rounds later. This means that, although the rational workers are partially available, the mechanism is able to reinforce them to an honest behavior eventually.

Acknowledgments: Supported in part by MINECO grant TEC2014- 55713-R, Regional Government of Madrid (CM) grant Cloud4BigData (S2013/ICE-2894, co- funded by FSE & FEDER), NSF of China grant 61520106005, EC H2020 grants ReCred and NOTRE, U. of Cyprus (ED-CG2015), the MECD grant FPU2013-03792 and Kean University RTR2016.

References

[1] Abraham, I., Dolev, D., Gonen, R., Halpern, J.: Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In Proc. of ACM PODC 2006, pp. 53–62.

[2] Allen, B.: The Einstein@home project (2014). `http://einstein.phys.uwm.edu`.

[3] *Amazon's Mechanical Turk* (2014), https://www.mturk.com.
[4] Anderson, D.P.: BOINC: A system for public-resource computing and storage. In Proc.of the 5th IEEE/ACM International Workshop on Grid Computing, pp. 4–10 (2004).
[5] Anderson, D.P.: Volunteer computing: the ultimate cloud. ACM Crossroads 16(3):7–10 (2010).
[6] Anderson, D.P.: BOINC reputation (2014), http://boinc.berkeley.edu/trac/wiki/AdaptiveReplication.
[7] Anderson, D.P.: BOINC (2016), http://boinc.berkeley.edu/.
[8] Bush, R.R., Mosteller, F.: Stochastic models for learning (1955).
[9] Christoforou, E., Fernández Anta, A., Georgiou, C., Mosteiro, M.A.: Algorithmic mechanisms for reliable master-worker internet-based computing. IEEE Trans. Computers 63(1):179–195 (2014).
[10] Christoforou, E., Fernández Anta, A., Georgiou, C., Mosteiro, M.A., Sánchez, A.: Applying the dynamics of evolution to achieve reliability in master-worker computing. Concurrency and Computation: Practice and Experience 25(17):2363–2380 (2013).
[11] Christoforou, E., Fernández Anta, A., Georgiou, C., Mosteiro, M.A., Sánchez, A.: Reputation-based mechanisms for evolutionary master-worker computing. In Proc. of OPODIS 2013, pp. 98–113.
[12] Estrada, T., Taufer, M., Anderson, D.P.: Performance prediction and analysis of BOINC projects: An empirical study with EMBOINC. J. of Grid Computing 7(4):537–554 (2009).
[13] Fernández Anta, A., Georgiou, C., López, L., Santos, A.: Reliable Internet-based master-worker computing in the presence of malicious workers. Par. Proc. Letters 22(01) (2012).
[14] Fernández Anta, A., Georgiou, C., Mosteiro, M.A.: Designing mechanisms for reliable Internet-based computing. In Proc. of IEEE NCA 2008, pp. 315–324.
[15] Golle, P., Mironov, I.: Uncheatable distributed computations. Topics in Cryptology-CT-RSA 2001, pp. 425–440.
[16] Heien, E.M., Anderson, D.P., Hagihara, K.: Computing low latency batches with unreliable workers in volunteer computing environments. J. of Grid Computing 7(4):501–518 (2009).
[17] Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision support systems 43(2):618–644 (2007).
[18] Kondo, D., Araujo, F., Malecot, P., Domingues, P., Silva, L.M., Fedak, G., Cappello, F.: Characterizing result errors in Internet desktop grids. In Euro-Par 2007, pp. 361–371.
[19] Konwar, K.M., Rajasekaran, S., Shvartsman, A.A.: Robust network supercomputing with malicious processes. In Proc. of DISC 2006, pp. 474–488.
[20] Korpela, E., Werthimer, D., Anderson, D.P., Cobb, J., Lebofsky, M.: SETI@home: massively distributed computing for SETI. Computing in Sci. & Eng. 3(1):78–83 (2001).
[21] Sarmenta, L.F.: Sabotage-tolerance mechanisms for volunteer computing systems. Future Generation Computer Systems 18(4):561–572 (2002).
[22] Shneidman, J., Parkes, D.C.: Rationality and self-interest in peer to peer networks. Peer-to-Peer Systems II, pp. 139–148 (2003).
[23] Smith, J.M.: Evolution and the Theory of Games. Cambridge university press (1982).
[24] Sonnek, J., Chandra, A., Weissman, J.B.: Adaptive reputation-based scheduling on unreliable distributed infrastructures. IEEE PDS 18(11):1551–1564 (2007).
[25] Szepesvári, C.: Algorithms for reinforcement learning. Synthesis Lectures on Artificial Intelligence and Machine Learning 4(1):1–103 (2010).
[26] Taufer, M., Anderson, D.P., Cicotti, P., Brooks III, C.L.: Homogeneous redundancy: a technique to ensure integrity of molecular simulation results using public computing. In Proc. of IEEE IPDPS 2005.
[27] Yurkewych, M., Levine, B.N., Rosenberg, A.L.: On the cost-ineffectiveness of redundancy in commercial P2P computing. In Proc. of ACM CCS 2005, pp. 280–288.