Achieving Reliability in Master-Worker Computing via Evolutionary Dynamics

Evgenia Christoforou[1], Antonio Fernández Anta[2], Chryssis Georgiou[1], Miguel A. Mosteiro[3], and Angel (Anxo) Sánchez[4]

[1] University of Cyprus, Nicosia, Cyprus
[2] Institute IMDEA Networks, Madrid, Spain
[3] Rutgers University, Piscataway, NJ, USA & Universidad Rey Juan Carlos, Madrid, Spain
[4] Universidad Carlos III de Madrid, Madrid, Spain & BIFI Institute, Zaragoza, Spain

Abstract. We consider Internet-based Master-Worker computations, like SETI@home, where a master process sends tasks, across the Internet, to worker processes; workers execute and report back some result. However, these workers are not trustworthy and it might be at their best interest to report incorrect results. In such master-worker computations, the behavior and the best interest of the workers might change over time.
We model such computations using evolutionary dynamics and we study the conditions under which the master can reliably obtain tasks results. In particular, we develop and analyze an algorithmic mechanism based on reinforcement learning to provide workers with the necessary incentives to eventually become truthful. Our analysis identifies the conditions under which truthful behavior can be ensured, and bounds the expected convergence time to that behavior. The analysis is complemented with simulations.

Keywords: Performing tasks, Internet-based Computing, Evolutionary dynamics, Reinforcement learning, Algorithmic mechanism design.

1 Introduction

Motivation: As an alternative to expensive supercomputing parallel machines, Internet is a feasible computational platform for processing complex computational jobs. Several Internet-based applications operate on top of this global computation infrastructure. Examples are volunteer-based "@home" projects [5] such as SETI [19] and profit-seeking computation platforms such as Amazon's Mechanical Turk [4].

Although the potential is great, the use of Internet-based computing is limited by the untrustworthy nature of the platform's components [5,15,16]. In SETI, for example, there is a machine, call it the *master*, that sends tasks, across the Internet, to volunteers' computers, call them *workers*. These workers execute and report back some result. However, these workers may not be trustworthy and it might be at their best interest to report incorrect results; that is, workers, or their owners, can be viewed as *rational* [2, 15, 28]. In SETI, the master attempts to minimize the impact of these bogus results by assigning the same task to several workers and comparing their outcomes (that is, redundant task allocation is employed [5, 27]).

Prior work [12,13,33] has shown that it is possible to design algorithmic mechanisms with reward/punish schemes so that the master can reliably obtain correct task results. We view these mechanisms as one-shot in the following sense: In a round, the master sends a task to be computed to a collection of workers, and the mechanism, using auditing and reward/punish schemes guarantees (with high probability) that the master gets the correct task result. For another task to be computed, the process is repeated (with the same or different collection of workers) but without taking advantage of the knowledge gained.

Given a long running computation (such as SETI-like master-worker computations), it can be the case that the best interests, and hence the behavior of the workers, might change over time. So, one wonders: Would it be possible to design a mechanism for performing many tasks, over the course of a possibly infinite computation, that could positively exploit the repeated interaction between a master and the same collection of workers?

Our approach: In this work we provide a positive answer to the above question. To do so, we introduce the concept of *evolutionary dynamics* under the biological and social perspective and relate them to Internet-based master-worker task computing. More specifically, we employ *reinforcement learning* [8,29] to model how system entities or learners interact with the environment to decide upon a strategy, and use their experience to select or avoid actions according to the consequences observed. Positive payoffs increase the probability of the strategy just chosen, and negative payoffs reduce this probability. Payoffs are seen as parameterizations of players' responses to their experiences. Empirical evidence [7, 9, 26] suggests that reinforcement learning is more plausible with players that have information only on the payoffs they receive; they do not have knowledge of the strategies involved. This model of learning fits nicely to our master-worker computation problem: the workers have no information about the master and the other workers' strategies and they don't know the set of strategies that led to the payoff they receive. The workers have only information about the strategies they choose at each round of the evolution of the system and their own received payoffs. The master also has minimal information about the workers and their intentions (to be truthful or not). Thus, we employ reinforcement learning for both the master and the workers in an attempt to build a reliable computational platform.

Our contributions:

- We formulate and study the dynamics of the evolution of Internet-based master-worker computations through reinforcement learning.
- We develop and analyze a mechanism based on reinforcement learning to be used by the master and the workers. In particular, in each round, the master allocates a task to the workers and decides whether to audit or not their responses with a certain probability $p_{\mathcal{A}}$. Depending on whether it audits or not, it applies a different reward/punish scheme, and adjusts the probability $p_{\mathcal{A}}$ for the next round (a.k.a. the next task execution). Similarly, in a round, each worker i decides whether it will truthfully compute and report the correct task result, or it will report an incorrect result, with a certain probability p_{Ci}. Depending on the success or not of its decision, measured by the increase or the decrease of the worker's *utility*, the worker adjusts probability p_{Ci} for the next round.
- We show necessary and sufficient conditions under which the mechanism ensures *eventual correctness*, that is, we show the conditions under which, after some finite number of rounds, the master obtains the correct task result in every round, with minimal auditing, while keeping the workers satisfied (w.r.t. their utility). Eventual correctness can be viewed as a form of *Evolutionary Stable Strategy* [10, 14] as studied in Evolutionary Game Theory [32]: even if a "mutant" worker decides to change its strategy to cheating, it will soon be brought back to an honest strategy.
- Finally, we show that our mechanism, when adhering to the above-mentioned conditions, reaches eventual correctness quickly. In particular, we show analytically, probabilistic bounds on the convergence time, as well as bounds on the expected convergence time. Our analysis is complemented with simulations.

Background and Related Work: Evolutionary dynamics were first studied in evolutionary biology, as a tool to studying the mathematical principles according to which life is evolving. Many fields were inspired by the principles of evolution dynamics such as evolutionary sociology, evolutionary economics and artificial intelligence (multi-agent systems). Our work is inspired by dynamics of evolution as a mean to model workers adaptation to a truthful behavior.

The dynamics of evolution have mainly been studied under the principles of Evolutionary Game Theory (EGT) [21, 32]. In EGT the concept of evolutionarily stable strategy (ESS) is used [10, 14, 32]. A strategy is called evolutionary stable if, when the whole population is using this strategy, any group of invaders (mutants) using a different strategy will eventually die off over multiple generations (evolutionary rounds). It is shown [14] that an ESS is a Nash Equilibrium, but the reverse is not true.

While evolution operates on the global distribution of strategies within a given population, reinforcement learning [29] operates on the individual level of distribution over strategies of each member of the population. There are several models of reinforcement learning (see [17, 29] for details). A well-known model is the Bush and Mosteller's model [8]. This is an aspiration-based reinforcement learning model where negative effects on the probability distribution over strategies are possible, and learning does not fade with time. The player's adapt by comparing their experience with an *aspiration* level. In our work we adapt this reinforcement learning model and we consider a simple aspiration scheme where aspiration is fixed by the workers and does not change during the evolutionary process.

Phelps, McBurney and Parsons [24] discusses the concept of Evolutionary Mechanism Design. The evolutionary mechanism has a continues interaction and feedback from the current mechanism, as opposed to classical mechanism design than when the mechanism is introduced in the system, it remains in the same Nash Equilibrium forever. In some way, our mechanism can be seen as an evolutionary mechanism, since the probability of auditing of the master and the probability of cheating of the workers, change, which is similar to changing the mechanism.

Several works that apply game theory concepts to distributed computing exist, including works on Internet routing, resource/facility location and sharing, containment of viruses spreading, secret sharing, P2P services and task computations. For more discussion on the connection between game theory and distributed computing see the survey Abraham, Alvisi and Halpern [1] and the book by Nisan et al [23].

The problem of achieving reliability in master-worker computations has been studied under two different views: from a traditional distributed computing view [11, 18, 27] and from a game-theoretic view [12, 13, 33]. Under the first view, the workers are classified as either *malicious* or *altruistic*, based on a predefined behavior: malicious workers always report incorrect results to the master, while altruistic workers always compute and truthfully report the correct task result. To tackle the problem, typical malicious-tolerant voting protocols have been designed. Under the game-theoretic view, workers are assumed to be *rational* [2, 15] and act upon their own *self-interest*: the decide to be honest or cheat depending on which strategy increases their utility. To tackle the problem, algorithmic mechanisms [2, 22] using reward/punish schemes are designed, in an attempt to provide incentives to rational workers to act correctly. As already mentioned, these works [12,13,33] do not take into consideration the evolution of the computation. The mechanisms involve an one-shot interaction between the master and the workers, and for many tasks this interaction is repeated without taking advantage the experience that can be gained over prior interactions.

Distributed computation in the presence of selfishness was studied within the scope of combinatorial agencies in Economics [6]. The basic model considered is a combinatorial variant of the classical principal-agent problem [20]: A master (principal) must motivate a collection of workers (agents) to exert costly effort on the master's behalf, but the workers' actions are hidden from the master. Instead of focusing on each worker's actions, the focus is on complex combinations of the efforts of the workers that influence the outcome. The principal-agent approach is mostly considered with designing contracts between the principal and the workers that allow the principal to get the most out of the workers without knowing a priori what their actual capabilities are. An evolutionary version of the principal-agent problem has been considered [25]. The main difference with our master-worker framework is that the worker's actions cannot be viewed as *hidden*. The master receives a response by each worker and it is aware that either the worker has truthfully performed the task or not.

A line work, initiated by Aiyer et al. [3], considers the BAR model to reason about systems with Byzantine (malicious), Altruistic, and Rational participants. They also introduce the notion of a protocol being BAR-tolerant, that is, the protocol is resilient to both Byzantine faults and rational manipulation. It would be interesting to consider an *evolutionary* BAR-Tolerant model, where the system parameters would take advantage of the iterative communication and evolve though a learning process.

A detailed account of related work can be found in [31].

2 Model and Definitions

Master-Worker Framework: We consider a distributed system consisting of a master processor that assigns, over the Internet, computational tasks to a set of n workers (w.l.o.g., we assume that n is odd). In particular, the computation is broken into rounds, and in each round the master sends a task to the workers to compute and return the task result. The master, based on the workers' replies, must decide on the value it believes is the correct outcome of the task in the same round. The tasks considered in this work are assumed to have a unique solution; although such limitation reduces the scope of application of the presented mechanism [30], there are plenty of computations where the correct solution is unique: e.g., any mathematical function.

Following Abraham et al. [2], and Shneidman and Parkes [28], we assume that workers are *rational*, that is, they are selfish in a game-theoretic sense and their aim is to maximize their benefit (utility) under the assumption that other workers do the same. In the context of this paper, a worker is *honest* in a round, when it truthfully computes and returns the task result, and it *cheats* when it returns some incorrect value. So, a worker decides to be honest or cheat depending on which strategy maximizes its utility. We denote by p^r_{Ci} the probability of a worker i cheating in round r. This probability is not fixed, the worker adjusts it over the course of the computation.

While it is assumed that workers make their decision individually and with no coordination, it is assumed that all the workers that cheat in a round return the same incorrect value (as done, for example, in [27], [11] and [12]). This yields a worst case scenario (and hence analysis) for the master with respect to obtaining the correct result using mechanisms where the result is the outcome of voting; it subsumes models where cheaters do not necessarily return the same answer. (In some sense, this can be seen as a cost-free, weak form of collusion.)

Auditing, Payoffs, Rewards and Aspiration: To "persuade" workers to be honest, the master employs, when necessary, *auditing* and *reward/punish* schemes. The master, in a round, might 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 $p_{\mathcal{A}}$ the probability of the master auditing the responses of the workers. The master can change this auditing probability over the course of the computation. However, unless otherwise stated, we assume that there is a value $p^{min}_{\mathcal{A}} > 0$ so that at all times $p_{\mathcal{A}} \geq p^{min}_{\mathcal{A}}$.

Furthermore, the master can reward and punish workers, which can be used (possibly combined with auditing) to encourage workers to be honest. When the master audits, it can accurately reward and punish workers. When the master does not audit, it decides on the majority of the received replies, and it rewards only the majority. We refer to this as the $\mathcal{R}_{\mathrm{m}}$ reward scheme.

The payoff parameters considered in this work are detailed in Table 1. All these parameters are non-negative. Note that the first letter of the parameter's name identifies whose parameter it is. M stands for master and W for worker. Then, the second letter gives the type of parameter. P stands for punishment, C for cost, and B for benefit. Observe that there are different parameters for the reward $WB_{\mathcal{Y}}$ to a worker and the cost $MC_{\mathcal{Y}}$ of this reward to the master. This models the fact that the cost to the master might be different from the benefit for a worker.

We assume that, in every round, a worker i has an *aspiration* a_i, that is, the minimum benefit it expects to obtain in a round. In order to motivate the worker to participate in the computation, the master must ensure that $WB_{\mathcal{Y}} \geq a_i$; in other words, the worker has the potential of its aspiration to be covered. We assume that the master knows the aspirations. This information can be included, for example, in a contract the master and the worker agree on, prior to the start of the computation.

$WP_{\mathcal{C}}$	worker's punishment for being caught cheating
$WC_{\mathcal{T}}$	worker's cost for computing the task
$WB_{\mathcal{Y}}$	worker's benefit from master's acceptance
$MP_{\mathcal{W}}$	master's punishment for accepting a wrong answer
$MC_{\mathcal{Y}}$	master's cost for accepting the worker's answer
$MC_{\mathcal{A}}$	master's cost for auditing worker's answers
$MB_{\mathcal{R}}$	master's benefit from accepting the right answer

Table 1. Payoffs

Note that, among the parameters involved, we assume that the master has the freedom of choosing $WB_{\mathcal{Y}}$ and $WP_{\mathcal{C}}$; by tuning these parameters and choosing n, the master can achieve the goal of eventual correctness. All other parameters can either be fixed because they are system parameters or may also be chosen by the master (except the aspiration, which is a parameter set by each worker).

Eventual Correctness: The goal of the master is to eventually obtain a reliable computational platform. In other words, after some finite number of rounds, the system must guarantee that the master obtains the correct task results in every round with probability 1. We call such property *eventual correctness*.

3 Algorithmic Mechanism

We now detail the algorithms run by the Master and the workers.

Master's Algorithm: The master's algorithm begins by choosing the initial probability of auditing. After that, at each round, the master sends a task to all workers and, when all answers are received (a reliable network is assumed), the master audits the answers with probability $p_{\mathcal{A}}$. In the case the answers are not audited, the master accepts the value contained in the majority of answers and continues to the next round with the same probability of auditing. In the case the answers are audited, the value $p_{\mathcal{A}}$ of the next round is reinforced (i.e., modified according to the outcome of the round). Then, the master rewards/penalizes the workers appropriately. The master initially has scarce or no information about the environment (e.g., workers initial p_C). The initial probability of auditing will be set according to the information the master possesses. For example if it has no information about the environment, a safe approach may be to initially set $p_{\mathcal{A}} = 0.5$.

Observe that, when the answers are not audited, the master has no information about the number of cheaters in the round. Thus, the probability $p_{\mathcal{A}}$ remains the same as in the previous round. When the answers are audited, the master can safely extract the number of cheaters. Then, the master adapts the auditing probability $p_{\mathcal{A}}$ according to this number. (We denote by $cheaters(r)$ the number of cheaters in round r.) Observe that the algorithm guarantees $p_{\mathcal{A}} \geq p_{\mathcal{A}}^{min}$. This, combined with the property $p_{\mathcal{A}}^{min} > 0$ will prevent the system to fall in a permanent set of "bad" states where $p_{\mathcal{A}} = 0$ and $p_C > 0$. A discount factor, which we call *tolerance* and denote by τ, expresses the master's tolerable ratio of cheaters (typically, we will assume $\tau = 1/2$). Hence, if the proportion of cheaters is larger than τ, $p_{\mathcal{A}}$ will be increased, and otherwise, $p_{\mathcal{A}}$ will be decreased. The amount by which $p_{\mathcal{A}}$ changes depends on the difference between these values, modulated by a *learning rate* α_m. 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: The workers' algorithm begins with each worker i deciding an initial probability of cheating p_{Ci}. At each round, each worker receives a task from the master and, with probability $1 - p_{Ci}$ calculates the task, and replies to the master with the correct answer. If the worker decides to cheat, it fabricates an answer, and sends the incorrect response to the master. We use a flag S_i to model the decision of a worker i to cheat or not. After receiving its payoff (detailed in the analysis section), each worker i changes its p_{Ci} according to the payoff Π_i received, the chosen strategy S_i, and its aspiration a_i. Observe that the workers' algorithm guarantees $0 \leq p_{Ci} \leq 1$. The workers have a learning rate α_w.

Algorithm 1 Master's Algorithm

$p_{\mathcal{A}} \leftarrow x$, *where* $x \in [p_{\mathcal{A}}^{min}, 1]$
for $r \leftarrow 1$ **to** ∞ **do**
  send *a task* T *to all workers in* W
  upon *receiving all answers* **do**
    audit the answers with probability $p_{\mathcal{A}}$
    if *the answers were not audited* **then**
      accept the majority
    else
      $p_{\mathcal{A}} \leftarrow \min\{1, \max\{p_{\mathcal{A}}^{min}, p_{\mathcal{A}} + \alpha_m(\frac{cheaters(r)}{n} - \tau)\}\}$
    end if
    give the appropriate payoff Π_i *to each worker* $i \in W$
end for

Algorithm 2 Algorithm for Worker i

$p_{Ci} \leftarrow y$, *where* $y \in [0, 1]$
for $r \leftarrow 1$ **to** ∞ **do**
  receive *a task* T *from the master*
  set $S_i \leftarrow -1$ *with probability* p_{Ci}, *and*
    $S_i \leftarrow 1$ *otherwise*
  if $S_i = 1$
    then $\sigma \leftarrow compute(T)$
  else $\sigma \leftarrow$ *arbitrary solution*
  send *response* σ *to the master*
  get payoff Π_i
  $p_{Ci} \leftarrow \max\{0, \min\{1, p_{Ci} - \alpha_w(\Pi_i - a_i)S_i\}\}$
end for

We assume that all workers have the same learning rate, that is, they learn in the same manner (see also the discussion in [29]; the learning rate is called step-size there); note that our analysis can be adjusted to accommodate also workers with different learning rates.

4 Analysis

We now analyze the mechanism, which is composed of the Master's and the workers' algorithms presented in the previous section. We first model the evolution of the mechanism as a Markov Chain, and then we prove necessary and sufficient conditions for achieving eventual correctness. Then, we provide analytical evidence that convergence to eventual correctness can be reached rather quickly. Observe in Algorithms 1 and 2 that there are a number of variables that may change in each round. We will denote the value of a variable X after a round r with a superindex r as X^r.

4.1 The Mechanism as a Markov Chain

We analyze the evolution of the master-workers system as a Markov chain. In order to do that, we define first the set of states and the transition function as follows.

Let the state of the Markov chain be given by the vector of probabilities $(p_{\mathcal{A}}, p_{C1}, p_{C2}, \ldots, p_{Cn})$. Then, we denote the state after round r by $(p_{\mathcal{A}}^r, p_{C1}^r, p_{C2}^r, \ldots, p_{Cn}^r)$. Observe from Algorithms 1 and 2 that any state $(p_{\mathcal{A}}, p_{C1}, p_{C2}, \ldots, p_{Cn})$ in which $p_{\mathcal{A}} \in [p_{\mathcal{A}}^{min}, 1]$, and $p_{Ci} \in [0, 1]$ for each worker i, is a possible initial state of the Markov chain. The workers' decisions, the number of cheaters, and the payoffs in round r are the stochastic outcome of the probabilities used in round r. Then, restricted to $p_{\mathcal{A}}^r \in [p_{\mathcal{A}}^{min}, 1]$ and $p_{Ci}^r \in [0, 1]$, we can describe the transition function of the Markov chain in detail. For each subset of workers $F \subseteq W$, $P(F) = \prod_{j \in F} p_{Cj}^{r-1} \prod_{k \notin F}(1 - p_{Ck}^{r-1})$ is the probability that the set of cheaters is exactly F in round r. Then, we have the following.

- With probability $p_{\mathcal{A}}^{r-1} \cdot P(F)$, the master audits when the set of cheaters is F, and then,
  (0) the master updates $p_{\mathcal{A}}$ as $p_{\mathcal{A}}^r = p_{\mathcal{A}}^{r-1} + \alpha_m(|F|/n - \tau)$, and
  (1) each worker $i \in F$ updates p_{Ci} as $p_{Ci}^r = p_{Ci}^{r-1} - \alpha_w(a_i + WP_{\mathcal{C}})$,
  (2) each worker $i \notin F$ updates p_{Ci} as $p_{Ci}^r = p_{Ci}^{r-1} + \alpha_w(a_i - (WB_{\mathcal{Y}} - WC_{\mathcal{T}}))$.
- With probability $(1 - p_{\mathcal{A}}^{r-1})P(F)$, the master does not audit when F is the set of cheaters. Then, the master does not change $p_{\mathcal{A}}$ and the workers update p_{Ci} as follows. For each $i \in F$,
  (3) if $|F| > n/2$ then $p_{Ci}^r = p_{Ci}^{r-1} + \alpha_w(WB_{\mathcal{Y}} - a_i)$,
  (4) if $|F| < n/2$ then $p_{Ci}^r = p_{Ci}^{r-1} - \alpha_w \cdot a_i$,

and for each $i \notin F$,
(5) if $|F| > n/2$ then $p^r_{Ci} = p^{r-1}_{Ci} + \alpha_w(a_i + WC_{\mathcal{T}})$,
(6) if $|F| < n/2$ then $p^r_{Ci} = p^{r-1}_{Ci} + \alpha_w(a_i - (WB_{\mathcal{Y}} - WC_{\mathcal{T}}))$.

The following terminology will be used throughout. Let a *covered worker* be one that is paid at least its aspiration a_i and the computing cost $WC_{\mathcal{T}}$. In any given round r, let an *honest worker* be one for which $p^{r-1}_C = 0$. Let an *honest state* be one where the *majority* of workers are honest. Let an *honest set* be any set of honest states. We refer to the opposite cases as *uncovered worker*, *cheater worker* ($p^{r-1}_C = 1$), *cheat state*, and *cheat set* respectively.

4.2 Conditions for Eventual Correctness

In this section we show the conditions under which the system can guarantee eventual correctness. We begin with some terminology. Let a set of states S be called *closed* if, once the chain is in any state $s \in S$, it will not move to any state $s' \notin S$. (A singleton closed set is called an *absorbing* state.) For any given set of states S, we say that the chain *reaches* (resp. *leaves*) the set S if the chain reaches some state $s \in S$ (resp. reaches some state $s \notin S$).

In order to show eventual correctness, we must show eventual convergence to a closed honest set. Thus, we need to show (i) that there exists at least one such closed honest set, (ii) that all closed sets are honest, and (iii) that one honest closed set is reachable from any initial state. In the following we refer to lemmata left to the full version of this paper for brevity. Lemma 1 in [31] shows that, if $p_A = 0$ some cheat set is closed. Given (ii), the necessity of $p^{min}_A > 0$ is motivated by this claim. Hence, $p_A > 0$ is assumed for the rest of the analysis. Lemma 2 in [31] shows that, if the majority of workers is uncovered, no honest set is closed. Given (i), the necessity of a covered majority is motivated. Hence, it is assumed that the majority of workers are covered for the rest of the analysis. Lemma 3 in [31] shows that the honest set including all the states in which all covered workers are honest is closed, which proves (i). Lemma 4 in [31] shows that any honest set where some covered worker is not honest is not closed, and Lemma 5 in [31] shows that any set that is not honest is not closed. Together, they prove (ii), and also (iii) because, if only honest sets are closed, there is a way of going from non-honest sets to one of them. The overall result is established in Theorem 1.

The following theorem shows that there is a positive probability of reaching some state after which correctness can be guaranteed, as long as for a chosen majority of workers, the payment is enough to cover their aspiration and cost of performing the task.

Theorem 1. *If $p_{\mathcal{A}} > 0$ then, in order to guarantee with positive probability that, after some finite number of rounds, the system achieves eventual correctness, it is* **necessary** *and* **sufficient** *to set* $\boxed{WB_{\mathcal{Y}} \geq a_i + WC_{\mathcal{T}}}$ *for all $i \in Z$ in some set $Z \subseteq W$ such that $|Z| > n/2$.*

Proof. Direct from Lemmas 3 to 5 in the full version of this paper [31].

Remark: From Algorithm 1 it is easy to see that once the closed set $S = \{(p_{\mathcal{A}}, p_{C1}, \ldots, p_{Cn}) | \forall w \in Z : p_{Cw} = 0\}$ is reached, eventually $p_{\mathcal{A}} = p^{min}_{\mathcal{A}}$ and stays such forever.

4.3 Convergence Time

Theorem 1 shows necessary and sufficient conditions to achieve eventual correctness. However, in order to have a practical system, it is necessary to bound the time taken to achieve it, which we call the *convergence time*. In other words, starting from any initial state, we want to compute the number of rounds that takes to the Markov chain to reach an honest closed set. In this section, we show bounds on the convergence time.

Expected Convergence Time: Let C be the set of all covered workers. We assume, as required by Theorem 1, that $|C| > n/2$. From transitions (1) and (2) in the Markov chain definition, it can be seen that it is enough to have a consecutive sequence of $1/(\alpha_w \min\{ WB_{\mathcal{Y}} - a_i - WC_{\mathcal{T}}, WP_{\mathcal{C}} + a_i\}), \forall i \in C$, audits to enforce $p_C = 0$ for all covered workers. Which gives the following upper bound on the convergence time.

Theorem 2. *The expected convergence time is at most $\rho/(p_A^{min})^\rho$, where $\rho = 1/(\alpha_w \min_{i \in C}\{ WB_{\mathcal{Y}} - a_i - WC_{\mathcal{T}}, WP_{\mathcal{C}} + a_i\})$ and C is the set of covered workers.*

Proof. The expected convergence time is upper bounded by the expected time for ρ consecutive audits. Consider the time divided in phases of ρ rounds. Let a phase where the master audits in all rounds be called *successful*. The expected time for ρ consecutive audits, is at most the expected time for a successful phase. The probability of success in any given phase is at least $(p_A^{min})^\rho$. Consider the probability distribution of the number X of phases needed to have success, each with probability $(p_A^{min})^\rho$. This distribution is geometric and the expectation of X is $1/(p_A^{min})^\rho$. Given that each phase has ρ rounds, the claim follows.

The upper bound shown in Theorem 2 may be too pessimistic for certain values of the parameters. The following theorem provides a tighter bound under certain conditions.

Theorem 3. *Let us define, for each worker i, $dec_i \triangleq \alpha_w \min\{ WP_{\mathcal{C}} + a_i, WB_{\mathcal{Y}} - WC_{\mathcal{T}} - a_i\}$, and $inc_i \triangleq \alpha_w \max\{ WB_{\mathcal{Y}} - a_i, WC_{\mathcal{T}} + a_i\}$. Let C be the set of covered workers. If $p_A^{min} = \max_{i \in C}\{inc_i/(inc_i + dec_i)\} + \varepsilon$, for some $0 < \varepsilon < 1 - \max_{i \in C}\{inc_i/(inc_i + dec_i)\}$, the expected convergence time is $1/(\varepsilon \min_{i \in C} dec_i)$.*

Proof. Consider some fixed worker i. As a worst case, assume that $p_{Ci}^0 = 1$, the master audits always with probability p_A^{min}, and whenever the master does not audit p_{Ci} is increased. For any given round, it can be seen from the transition function that, in the worst case, p_{Ci} is decreased by dec_i when the master audits and increased by inc_i when the master does not audit. We want to compute the expected time to reach $p_{Ci} = 0$. Consider a potential function ϕ over the rounds that is updated as p_{Ci} but it is not bounded to $[0, 1]$. We claim that the expected number of rounds needed to get $\phi \leq 0$ is not less than the expected number of rounds to get $p_{Ci} = 0$. The reason is that if p_{Ci} is decreased more aggressively than ϕ in expectation, the value 0 is reached earlier. On the other hand, p_{Ci} is increased less aggressively than ϕ in expectation, given that it is bounded to 1. Then, less decreases will be needed later to reduce it to 0. We compute the expected number of rounds needed to get $\phi \leq 0$ as follows. We need $1/dec_i$ audits for each $1/inc_i$ non-audit rounds to compensate for the increase in potential. Setting $p_A^{min} = inc_i/(inc_i + dec_i)$ the master achieves at least that ratio in expectation for any time period. (Omitting that time is discrete for clarity.) Additionally, in order to compensate for the initial $\phi(0) = 1$, $1/dec_i$ additional audits are needed. Making $p_A^{min} = inc_i/(inc_i + dec_i) + \varepsilon$, for some $0 < \varepsilon < 1 - inc_i/(inc_i + dec_i)$, the expected convergence time is $1/(\varepsilon \cdot dec_i)$. Applying the same argument to all covered nodes, the claim follows.

The following corollary is derived from the previous theorem for a suitable scenario.

Corollary 1. *If $WP_{\mathcal{C}} + a_i \geq WB_{\mathcal{Y}} - WC_{\mathcal{T}} - a_i$ and $WB_{\mathcal{Y}} - a_i \leq WC_{\mathcal{T}} + a_i$, $\forall i \in C$, and if*

$$p_A^{min} = \frac{WC_{\mathcal{T}} + \max_{i \in C} a_i}{WB_{\mathcal{Y}}} + \varepsilon,$$

where C is the set of covered workers and $0 < \varepsilon < 1 - (WC_{\mathcal{T}} + \max_{i \in C} a_i)/WB_{\mathcal{Y}}$, then the expected convergence time is ρ/ε, where $\rho = 1/(\alpha_w(WB_{\mathcal{Y}} - WC_{\mathcal{T}} - \max_{i \in C} a_i))$.

Probabilistic Bound on the Number of Rounds for Convergence: We show now that, under certain conditions on the parameters of the system, it is possible to bound the probability to achieve convergence and the number of rounds to do so. Assume that $p_{\mathcal{A}}^0 > 0$. Since $p_{\mathcal{A}}$ is not changed unless the master audits, we have the following.

Lemma 1. *Let $p^0_{\mathcal{A}} = p > 0$. Then, the master audits in the first $\rho = \ln(1/\varepsilon_1)/p$ rounds with probability at least $1 - \varepsilon_1$, for any $\varepsilon_1 \in (0, 1)$.*

Proof. The master audits in the first ρ rounds with probability $1 - (1-p)^\rho \geq 1 - \exp(-\rho \cdot p) = 1 - \varepsilon_1$.

Let us assume that the system parameters are such that, for all workers i, $\alpha_w(WP_{\mathcal{C}} + a_i) \in [0, 1]$ and $\alpha_w(WB_{\mathcal{Y}} - WC_{\mathcal{T}} - a_i) \in (0, 1]$ (all workers are covered). Let us define $dec\_cheater \triangleq \alpha_w \min_i\{WP_{\mathcal{C}} + a_i\}$ and $dec\_honest \triangleq \alpha_w \min_i\{WB_{\mathcal{Y}} - WC_{\mathcal{T}} - a_i\}$. From transitions (1) and (2) we derive the following lemma.

Lemma 2. *Let r be a round in which the master audits, and F be the set of cheaters in round r. Then,*

$$p^r_{Ci} \leq 1 - \alpha_w(WP_{\mathcal{C}} + a_i) \leq 1 - dec\_cheater, \forall i \in F$$
$$p^r_{Cj} \leq 1 - \alpha_w(WB_{\mathcal{Y}} - WC_{\mathcal{T}} - a_j) \leq 1 - dec\_honest, \forall j \notin F$$

Let us denote the sum of all cheating probabilities before a round r as $P^{r-1} \triangleq \sum_i p^{r-1}_{Ci}$.

Lemma 3. *Let r be a round in which the master audits such that $P^{r-1} > n/3$. If $dec\_cheater \geq dec\_honest$ and $dec\_cheater + 3 \cdot dec\_honest \geq 8/3$, then $P^r \leq n/3$ with probability at least $1 - \exp(-n/96)$.*

Proof. Let F be the set of cheaters in round r. Then, using a Chernoff bound $Pr[|F| < (1-\delta)P^{r-1}] \leq \exp(-\delta^2 P^{r-1}/2)$, for any $\delta \in (0, 1)$. Then, since $P^{r-1} > n/3$, using $\delta = 1/4$, there are at least $(1 - \delta)P^{r-1} > n/4$ cheaters with probability at least $1 - \exp(-\delta^2 P^{r-1}/2) > 1 - \exp(-n/96)$. If that is the case, from Lemma 2 and $dec\_cheater \geq dec\_honest$, we have that

$$\begin{aligned} P^r &\leq n - |F| dec\_cheater - (n - |F|) dec\_honest \leq n - (n/4) dec\_cheater - (3n/4) dec\_honest \\ &= n(1 - (dec\_cheater + 3 \cdot dec\_honest)/4) \leq n/3. \end{aligned}$$

Let us now define $dec_i \triangleq \alpha_w \min\{a_i, WB_{\mathcal{Y}} - WC_{\mathcal{T}} - a_i\}$. Let, $dec \triangleq \min_i dec_i$. Assume $WP_{\mathcal{C}} \geq 0$ and $a_i \geq 0$, for all workers.

Lemma 4. *Consider a round r such that $P^{r-1} \leq n/3$. Then, with probability at least $1 - \exp(-n/36)$ each worker i has $p^r_{Ci} \leq \max\{0, p^{r-1}_{Ci} - dec\}$, and hence $P^r \leq n/3$.*

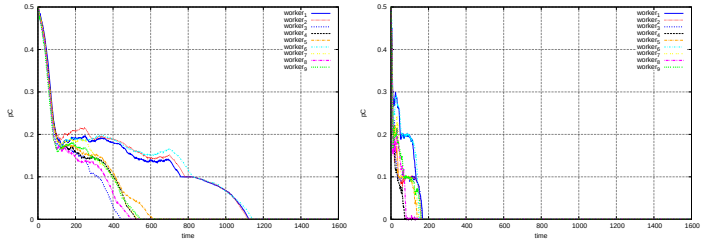
Proof. Using Chernoff, there is a majority of honest workers with probability at least

$$Pr[majority\ honest | P^{r-1} \leq n/3] \geq 1 - \exp(-(1/2)^2(n/3)/3) = 1 - \exp(-n/36).$$

It can be observed in Algorithm 2 that, if there is a majority of honest workers in a round r, then any worker i has $p^r_{Ci} \leq \max\{0, p^{r-1}_{Ci} - dec\}$, independently of whether the master audits. Hence the proof.

Theorem 4. *Assume $\alpha_w(WP_{\mathcal{C}} + a_i) \in [0, 1]$ and $\alpha_w(WB_{\mathcal{Y}} - WC_{\mathcal{T}} - a_i) \in (0, 1]$ for all workers i. (Observe that all workers are covered.) Let $dec\_cheater \triangleq \alpha_w \min_i\{WP_{\mathcal{C}} + a_i\}$, $dec\_honest \triangleq \alpha_w \min_i\{WB_{\mathcal{Y}} - WC_{\mathcal{T}} - a_i\}$, and $dec \triangleq \alpha_w \min_i\{a_i, WB_{\mathcal{Y}} - WC_{\mathcal{T}} - a_i\}$. If $p^0_{\mathcal{A}} = p > 0$, $dec\_cheater \geq dec\_honest$ and $dec\_cheater + 3 \cdot dec\_honest \geq 8/3$, then eventual convergence is reached in at most $\ln(1/\varepsilon_1)/p + 1/dec$ rounds, with probability at least $(1 - \varepsilon_1)(1 - \exp(-n/96))(1 - \exp(-n/36))^{1/dec}$, for any $\varepsilon_1 \in (0, 1)$.*

Proof. Consider the first round r in which the masters audits. From Lemma 1, r is in the first $\ln(1/\varepsilon_1)/p$ rounds with probability at least $1 - \varepsilon_1$. If so, either $P^{r-1} \leq n/3$ and also $P^r \leq n/3$ (from Algorithm 2 and the fact that the master audits in round r, P cannot increase in round r), or $P^{r-1} > n/3$. In this latter case, from Lemma 3, $P^r \leq n/3$ with probability at least $1 - \exp(-n/96)$. Then starting at round $r+1$, from Lemma 4, with probability at least $(1 - \exp(-n/36))^{1/dec}$, there are $1/dec$ consecutive rounds with majorities of honest workers. Since in each of these rounds the cheating probability of any worker decreases at least by dec (unless it is already zero), at the end of these rounds all workers have zero cheating probability.



(a) (b)

Fig. 1. Cheating probability for the workers as a function of time (number of rounds) for parameters $WB_{\mathcal{Y}} = 1$, $WP_{\mathcal{C}} = 0$, $WC_{\mathcal{T}} = 0.1$ and $a_i = 0.1$. (a) $\alpha = 0.01$; (b) $\alpha = 0.1$.

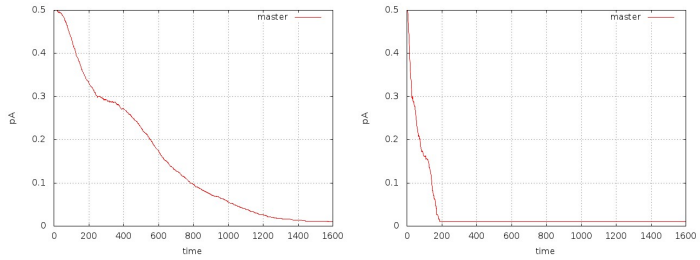
5 Simulations

In this section we complement the theoretical analysis with simulations. Our analytical upper bounds on convergence time correspond to worst case scenarios. Here we present simulations for a variety of parameter combinations likely to occur in practice. We have created our own simulation setup by implementing our mechanism (the master's and the workers' algorithms) using the C++ programming language. We have run our simulations on a PC with an Intel Core 2 Duo, 2.80GHz CPU, 4GB of RAM and Ubuntu 11.04 OS. Each depicted plot value represents the average over 10 executions of the implementation. We have simulated several scenarios for different parameter values (here we present selected results).

We choose sensible parameter values, likely to be encountered in real applications. In particular, the number of workers has been set to nine (an odd number to accommodate majority voting when the master does not audit). In systems like Seti@home [19] typically three-redundancy level is used, so in that context nine workers seems like an appropriate workforce. The initial cheating probability of each worker i is not known, and therefore we have set it at $p_{Ci} = 0.5$ as a reasonable assumption, product of our ignorance. Similarly, we have set $p_{\mathcal{A}} = 0.5$ as the master's initial probability of auditing. The minimum probability of cheating is set to be $p_{\mathcal{A}}^{min} = 0.01$ and tolerance $\tau = 0.5$, which means that the master will not tolerate a majority of cheaters. Beyond this intuition, we checked other values and found that for the range of parameters we studied tolerances lower than 0.6 without punishment and lower than 0.9 with punishment yield also convergence. For more details we refer to [31].

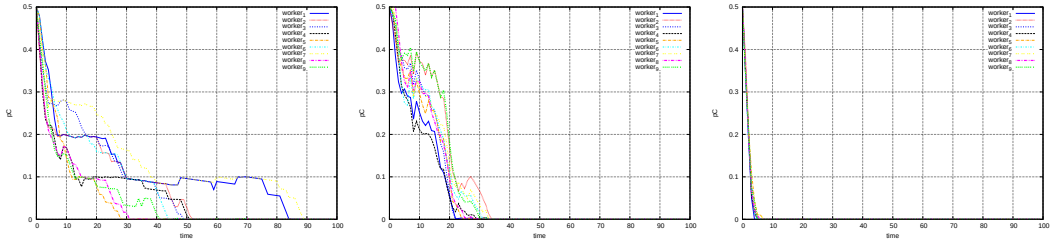
The payoffs for the workers are set using $WB_{\mathcal{Y}} \in \{1, 2\}$ as our normalizing parameter and we take $WP_{\mathcal{C}} \in \{0, 1, 2\}$ and $WC_{\mathcal{T}} = 0.1$ as realistic values (within the same order of magnitude as $WB_{\mathcal{Y}}$) to explore the effects of these choices. The aspiration is a parameter defined by the workers in an idiosyncratic manner; for simplicity, in these simulations we consider all workers having the same aspiration level $a_i = 0.1$. We have checked that when values are assigned randomly around some mean, the results are similar to those presented here, provided the variance is not very large. As for the values for the aspiration and of the workers' cost for computing the task $WC_{\mathcal{T}}$, they are such that the necessary conditions of Theorem 1 are satisfied and hence eventual convergence is reached. Finally, we consider the same learning rate for the master and the workers, i.e., $\alpha = \alpha_m = \alpha_w$. The learning rate, as discussed for example in [29] (called step-size there), for practical reasons it can be set to a small constant value; so we consider $\alpha \in \{0.1, 0.01\}$.

To begin with our examples, Figure 1 shows that convergence can be reached very quickly, in a few hundred rounds, even if no punishment is given to the workers caught cheating, and the number of workers and $WB_{\mathcal{Y}}$ are small. Note that even if all workers have the same aspiration level and begin with the same initial cheating probability, their evolution in time may be different from each other as it depends on the individual realizations of cheating. In Figure 1 we also notice that a slightly higher value of α can make the convergence time shorter. (As we argued before, the value of α can not be very high because the



(a) (b)

Fig. 2. Auditing probability for the master as a function of time (number of rounds) for parameters $WB_{\mathcal{Y}} = 1$, $WP_{\mathcal{C}} = 0$, $WC_{\mathcal{T}} = 0.1$ and $a_i = 0.1$. (a) $\alpha = 0.01$; (b) $\alpha = 0.1$.



(a) (b) (c)

Fig. 3. Cheating probability for the workers as a function of time (number of rounds) for parameters $\alpha = 0.1$, $a_i = 0.1$, $WB_{\mathcal{Y}} = 2$ and $WC_{\mathcal{T}} = 0.1$. (a) $WP_{\mathcal{C}} = 0$; (b) $WP_{\mathcal{C}} = 1$; (c) $WP_{\mathcal{C}} = 2$.

learning procedure will become unstable and p_C will bounce up and down without reaching convergence.) Similar conclusions can be drawn from Figure 2, where we can notice how quickly $p_{\mathcal{A}}$ drops to $p_{\mathcal{A}} = 0.01$, and also that $p_{\mathcal{A}}$ decreases in the same manner as p_C. Notice however, that $p_{\mathcal{A}}$ decreases at a slower rate; intuitively, this is to ensure that workers will not try and deviate from the desirable behavior.

Comparing Figure 1(b) with Figure 3(a) we observe that, for a specific set of parameter values, a larger $WB_{\mathcal{Y}}$ leads to a shorter convergence time. Interestingly, this observation points out to a trade-off between convergence time and the cost the master has for reaching faster convergence and maintaining it. In this way, the master could choose between different protocols estimating the cost of the auditing during the whole interval to convergence: less auditing leads to larger convergence times, so it is not clear in principle what is going to be optimal. Notice also that in the previous simulations only a positive reinforcement is applied to the workers (i.e., $WP_{\mathcal{C}} = 0$); now, from Figure 3 we can notice that the larger negative reinforcement we apply (i.e., $WP_{\mathcal{C}} = \{1, 2\}$) the faster the convergence time is. In fact, we may conclude that applying only a negative reinforcement is enough to have fast convergence.

References

[1] I. Abraham, L. Alvisi, and J.Y. Halpern. Distributed computing meets game theory: Combining insights from two fields. *ACM SIGACT News: Distributed Computing Column*, 42(2):69–76, 2011.

[2] I. Abraham, D. Dolev, R. Goden, and J.Y. Halpern. Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In *proc. of PODC 2006*, pp. 53–62.

[3] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J. Martin, and C. Porth. BAR fault tolerance for cooperative services. In *proc. of SOSP 2005*, pp. 45–58.

[4] Amazon's Mechanical Turk, `https://www.mturk.com`.

[5] D. Anderson. BOINC: A system for public-resource computing and storage. In *proc. of GRID 2004*, pp. 4–10.

[6] M. Babaioff, M. Feldman, and N. Nisan. Combinatorial agency. In *proc. of ACM EC 2006*, pp. 18–28.

[7] J. Bendor, D. Mookherjee and D. Ray. Aspiration-based reinforcement learning in repeated interaction games: An overview. *International Game Theory Review*, 3(2-3):159–174, 2001.
[8] R. R. Bush and F. Mosteller. *Stochastic Models for Learning*, Wiley, 1955.
[9] C. F. Camerer. Behavioral game theory: Experiments in strategic interaction. *Roundtable Series in Behavioral Economics*, 2003.
[10] D.Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*, Cambridge University Press, 2010.
[11] A. Fernández, Ch. Georgiou, L. Lopez, and A. Santos. Reliably executing tasks in the presence of untrusted processors. In *proc. of SRDS 2006*, pp. 39–50.
[12] A. Fernández Anta, Ch. Georgiou, and M. A. Mosteiro. Designing mechanisms for reliable Internet-based computing. In *proc. of NCA 2008*, pp. 315–324.
[13] A. Fernández Anta, Ch. Georgiou, and M. A. Mosteiro. Algorithmic Mechanisms for Internet-based Master-Worker Computing with Untrusted and Selfish Workers. In *proc. of IPDPS 2010*, pp. 1–11.
[14] M. C. Gintis. *Game Theory Evolving*, Princeton University Press, 2000.
[15] P. Golle and I. Mironov. Uncheatable distributed computations. In *proc. of CT-RSA 2001*, pp. 425–440.
[16] E.M. Heien, D.P. Anderson, and K. Hagihara. Computing low latency batches with unreliable workers in volunteer computing environments. *Journal of Grid Computing*, 7:501–518, 2009.
[17] L. P. Kaelbling, L. M. Littman, and A. W. Moore. Reinforcement learning: A survey. *J. Artif. Intell. Res.*, 4:237–285, 1996.
[18] K.M. Konwar, S. Rajasekaran, and A.A. Shvartsman. Robust network supercomputing with malicious processes. In *proc. of DISC 2006*, pp. 474–488.
[19] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Lebofsky. SETI@home: Massively distributed computing for SETI. *Computing in Science and Engineering*, 3(1):78–83, 2001.
[20] A. Mass-Colell, M. Whinton, and J. Green. *Microeconomic Theory*, Oxford University Press, 1995.
[21] J. Maynard Smith. *Evolution and the Theory of Games*, Cambridge University Press, 1982.
[22] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.
[23] N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani, editors. *Algorithmic Game Theory.* Cambridge University Press, 2007.
[24] S. Phelps, P. McBurney and S. Parsons. Evolutionary mechanism design: A review. *Journal of Autonomous Agents and Multi-Agent Systems*, 2010.
[25] D. Rose, T. R. Willemain. The principal-agent problem with evolutionary learning. *Computational and Mathematical Organization Theory*, 2:139–162, 1996.
[26] A. Roth and I. Erev. Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate Term. *Games and Economic Behavior*, 8:164–212, 1995.
[27] L. Sarmenta. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems*, 18(4):561–572, 2002.
[28] J. Shneidman and D.C. Parkes. Rationality and self-interest in P2P networks. In *IPTPS 2003*, pp. 139–148.
[29] C. Szepesvári. *Algorithms for Reinforcement Learning.* Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool publishers, 2010.
[30] M. Taufer, D. Anderson, P. Cicotti, and C. L. Brooks. Homogeneous redundancy: a technique to ensure integrity of molecular simulation results using public computing. In *proc. of IPDPS 2005.*
[31] Technical report of this work, TR-12-02, Dept. of Computer Science, University of Cyprus, February 2012. `http://www.cs.ucy.ac.cy/~chryssis/EvolMW-TR.pdf`
[32] J.W. Weibull. *Evolutionary Game Theory*, MIT Press, 1995.
[33] M. Yurkewych, B.N. Levine, and A.L. Rosenberg. On the cost-ineffectiveness of redundancy in commercial P2P computing. In *proc. of CCS 2005*, pp. 280–288.