UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

ACHIEVING RELIABILITY AND FAIRNESS IN ONLINE TASK COMPUTING ENVIRONMENTS

Autor
Evgenia Christoforou
IMDEA Networks Institute & Universidad Carlos III de Madrid

Co-Director
Antonio Fernández Anta
IMDEA Networks Institute

Co-Director & Tutor
Angel Sánchez
Universidad Carlos III de Madrid

DOCTORADO EN INGENIERÍA MATEMÁTICA
DEPARTAMENTO DE MATEMÁTICAS

Leganés (Madrid), Junio de 2017

UNIVERSIDAD CARLOS III DE MADRID

PH.D. THESIS

ACHIEVING RELIABILITY AND FAIRNESS IN ONLINE TASK COMPUTING ENVIRONMENTS

Author
Evgenia Christoforou
IMDEA Networks Institute & Universidad Carlos III de Madrid

Co-Director
Antonio Fernández Anta
IMDEA Networks Institute

Co-Director & Tutor
Angel Sánchez
Universidad Carlos III de Madrid

DOCTORATE IN MATHEMATICAL ENGINEERING
DEPARTMENT OF MATHEMATICS

Leganés (Madrid), June 2017

Achieving Reliability and Fairness in Online Task Computing Environments

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Prepared by
Evgenia Christoforou, IMDEA Networks Institute & Universidad Carlos III de Madrid

Under the advice of
Antonio Fernández Anta, IMDEA Networks Institute
Angel Sánchez, Universidad Carlos III de Madrid
Departamento de Matemáticas, Universidad Carlos III de Madrid

---

Date: Junio, 2017

Web/contact: evgenia.christoforou@imdea.org

This work has been supported by IMDEA Networks Institute and the Spanish Ministry of Education grant FPU2013-03792.

TESIS DOCTORAL

ACHIEVING RELIABILITY AND FAIRNESS IN ONLINE TASK COMPUTING ENVIRONMENTS

Autor
Evgenia Christoforou, IMDEA Networks Institute & Universidad Carlos III de Madrid

Co-Director
Antonio Fernández Anta, IMDEA Networks Institute

Co-Director & Tutor
Angel Sánchez, Universidad Carlos III de Madrid

Firma del tribunal calificador:

Presidente: Alberto Tarable

Vocal: Juan Julián Merelo Guervós

Secretario: José A. Cuesta

Calificación:

Leganés, 25 de Mayo de 2017

To my father, a great man and a great scientist...

Θάρρος

Στα δύσκολα εις την ζωήν, να μεν λιποτακτήσεις,
ούτε τζιαι την προσπάθειαν, στην μέση να αφήσεις.
Μες τη ζωήν την δύσκολην, ποττέ μεν σταματήσεις,
τζι' αν κάμεις τζιαι διακοπήν, πάλαι να ξεκινήσεις.
Αφού εγεννηθήκαμεν, μπροστά μας να θωρούμεν,
ούτε με σιόννια τζιαι βροσιές, πρέπει να σταματούμεν.
Τζι' αν αιστανθείς τον κίντυνον, εσού να μεν κολόσεις,
ούτε τζιαι τα πιστεύω σου, ποττέ σου να προδώσεις.

Θάρρος, Χαράλαμπος Χριστοφόρου, 2013

Acknowledgements

Before presenting this work, I would like to take a moment of the reader's time to thank the people who have supported and guided me through this process. First and foremost I would like to thank my parents, without them nothing of this would have been possible. They gave me a great opportunity in life by giving me an education, which is my "fortune" as they keep saying, and for that I am forever grateful. I would like to thank my mother, one of the strongest women I have ever met in my life, for her endless support, her endless careering and her endless love. Words are not enough to describe my father's support and guidance throughout my life. He is the kindest person I ever met and losing him was the hardest moment of my life. Dedicating this thesis to him is the least I can do for the person that taught me how to have courage in life. Moreover, I would like to thank my brother and my sister for being there and sharing every step of our life together.

Moving on, I would like to acknowledge my thesis directors. It is not an exaggeration to say that without them this work would not have been possible. They have helped mature as a scientist in every aspect, but one of the greatest lessons that Antonio gave me was to think out of the box, while Angel (Anxo) taught to have right-thinking. I would like to give a special thanks to Chryssis Georgiou, for being my first supervisor and showing me how to be a researcher. Moreover, I would like to thank my co-authors Miguel Mosteiro for the excellent collaboration all these years; Agustin Sántos for the amazing energy that he brings into the work; Kishori Konwar for sharing his amazing ideas and last but not least Nicolas Nicolaou for amount of time we spent working together and for the endless brainstorming.

A especial thanks goes out to four very dear friends of mine that are always there when I need them to support me and take my tension away. Voula with her infinite passion for life being always an optimist; Nuri always being a realist; Kalia with her absolute kindness; and Andri always there to listen and support. I would also like to thank Dario, Alessia, Gaia, Maurizio and Paolo for the great dinner company, enjoying together the Madrid tapas experience and Miguel for joining us to the adventurous salsa classes. I could not help but saying a great thanks to all my IMDEA current and former colleagues for sharing a coffee break and having a great team attitude. A particular thanks goes out to Pablo that although he is far he never stopped caring.

I would like to take a final moment to thank my partner in life Christian, that has been my pillar of strength. He has supported me through out this process not only by being my best friend but also by being my most sever critic. He has been encouraging me to continue through my toughest moments with every possible mean, without him this work would have never been completed.

Abstract

We consider online task computing environments such as volunteer computing platforms running on BOINC (e.g., SETI@home) and crowdsourcing platforms such as Amazon Mechanical Turk. We model the computations as an Internet-based task computing system under the master-worker paradigm. A master entity sends tasks across the Internet, to worker entities willing to perform a computational task. Workers execute the tasks, and report back the results, completing the computational round. Unfortunately, workers are untrustworthy and might report an incorrect result. Thus, the first research question we answer in this work is how to design a reliable master-worker task computing system. We capture the workers' behavior through two realistic models: (1) the "error probability model" which assumes the presence of altruistic workers willing to provide correct results and the presence of troll workers aiming at providing random incorrect results. Both types of workers suffer from an error probability altering their intended response. (2) The "rationality model" which assumes the presence of altruistic workers, always reporting a correct result, the presence of malicious workers always reporting an incorrect result, and the presence of rational workers following a strategy that will maximize their utility (benefit). The rational workers can choose among two strategies: either be honest and report a correct result, or cheat and report an incorrect result. Our two modeling assumptions on the workers' behavior are supported by an experimental evaluation we have performed on Amazon Mechanical Turk. Given the error probability model, we evaluate two reliability techniques: (1) "voting" and (2) "auditing" in terms of task assignments required and time invested for computing correctly a set of tasks with high probability. Considering the rationality model, we take an evolutionary game theoretic approach and we design mechanisms that eventually achieve a reliable computational platform where the master receives the correct task result with probability one and with minimal auditing cost. The designed mechanisms provide incentives to the rational workers, reinforcing their strategy to a correct behavior, while they are complemented by four reputation schemes that cope with malice. Finally, we also design a mechanism that deals with unresponsive workers by keeping a reputation related to the workers' response rate. The designed mechanism selects the most reliable and active workers in each computational round. Simulations, among other, depict the trade-off between the master's cost and the time the system needs to reach a state where the master always receives the correct task result. The second research question we answer in this work concerns the fair and efficient distribution of workers among the masters over multiple

computational rounds. Masters with similar tasks are competing for the same set of workers at each computational round. Workers must be assigned to the masters in a fair manner; when the master values a worker's contribution the most. We consider that a master might have a strategic behavior, declaring a dishonest valuation on a worker in each round, in an attempt to increase its benefit. This strategic behavior from the side of the masters might lead to unfair and inefficient assignments of workers. Applying renown auction mechanisms to solve the problem at hand can be infeasible since monetary payments are required on the side of the masters. Hence, we present an alternative mechanism for fair and efficient distribution of the workers in the presence of strategic masters, without the use of monetary incentives. We show analytically that our designed mechanism guarantees fairness, is socially efficient, and is truthful. Simulations favourably compare our designed mechanism with two benchmark auction mechanisms.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AMT Amazon Mechanical Turk

FEDoR Fair and Efficient Distribution of Resources

BOINC Berkeley Open-source software for volunteer computing

SETI Search for Extraterrestrial Intelligence

WCG World Community Grid

HIT Human Intelligence Task

EGT Evolutionary Game Theory

GT Game Theory

NE Nash Equilibrium

MD Mechanism Design

P2P Peer to Peer

CPU Central Processing Unit

QPQ Quid Pro Quo

whp with high probability

ECDF Empirical Cumulative Distribution Function

VCG Vickrey-Clark-Groves

GSP Generalized Second Price

GoF Goodness of Fit

PIT Probability Integral Transformation

Chapter 1

Introduction

The Internet is turning into a massive source of inexpensive computational power. Every entity connected over the Internet is a potential source not only of machine computational power but also of human intelligence. For this reason, numerous task computing environments [8, 23, 56, 102] have been designed to harvest this computational power. One of the most representative examples is volunteer computing initiatives, like the Search for Extraterrestrial Intelligence (SETI)@home [79] project, that is supported by the Berkeley Open-source software for volunteer computing (BOINC) [8]. Other representative examples are profit-seeking platforms, such as Amazon Mechanical Turk (AMT) [102]. The advantage of these online task computing platforms is the easy access and inexpensive computational power they provide.

We refer to the computation taking place in these environments as *Internet-based task computing*, and we model its components by means of a master-worker approach. A "master" entity has a set of tasks to perform, and instead of computing them locally, she sends these tasks across the Internet to "worker" entities that execute and report back some task result. A real computational environment may have multiple master entities with distinct set of tasks to perform.

Unfortunately, two crucial and challenging research problems prevent Internet-based task computing from reaching its full potential. First of all, computations carried out over the Internet can not be trusted due to the fact that workers can hide behind their anonymity or simply make mistakes. Moreover, whoever is requesting the computation may be lacking tools to verify the validity of the results. Thus, the results received from a worker can not be considered reliable, and the whole computational environment suffers from "unreliability" issues. From this first research challenge a complementary problem emerges, that is characterizing the nature of the workers. Being able to understand and model the workers behavior is an essential component to any designed solution. A second challenge arises from the fact that multiple masters may be competing at the same time for the same workers, and thus might experience issues arising from the inefficient allocation of the workers. This work provides a number of solutions addressing the problem of unreliability and inefficient resource allocation in Internet-based task computing setting. Figure 1.1 describes the two main research problems addressed in this work and how they

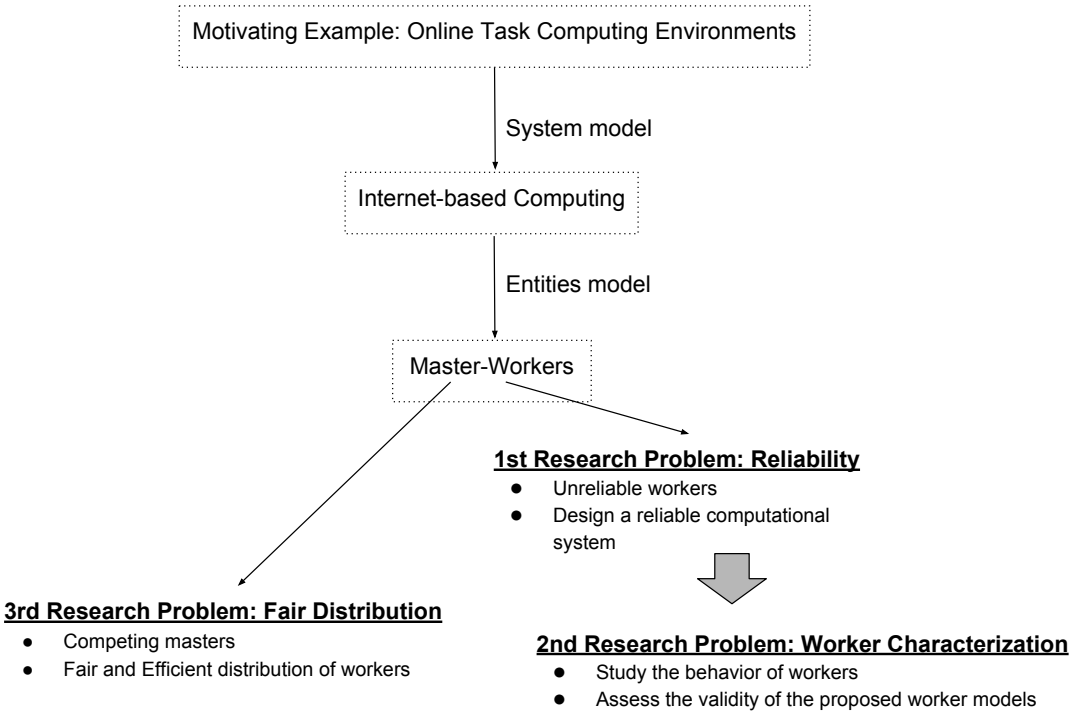


Figure 1.1: **System model and research problems**

arise from the task computing environment we consider.

Online Task Computing Platforms

This work focuses on computations taking place over the internet. By computations we refer to the processing of tasks run by machines or the processing of tasks performed through human intelligence. Thus, we classify computations based on the mean used for performing the task. Further more, we sub-classify these computation with respect to the payment received. Workers in online task computing might or might not receive monetary payments. We continue by describing four principal settings according to the above classification where online task computing platforms are present. In Table 1.1 we summarize the features of four types of task computations via the workers' attributes.

Volunteer computing has been greatly embraced by the scientific community that is always in need of cheap supercomputing power. End users engaged by the mission of the project are willing to contribute their machine's idle computational time. The majority of these volunteering projects use the BOINC platform [8], with SETI@home [79] being one of the most characteristic examples. Volunteer computing initiatives such as the one of IBM through the World Community Grid (WCG) [3] are able to bring together organizations dealing with health, poverty and sustainability with volunteers all over the Internet that want to put in a good use their idle processing power. Besides joining a project, with altruistic motives, to support a scientific goal, a worker

		Worker							
		Monetary Reward		Contribution		Motivation			
		Yes	No	Machine Power	Human Intelligence Power	Payment	Altruism	Enjoyment	Reputation
Task	Volunteer computing		x	x			x		x
	Crowdsourcing	x	x		x	x	x	x	x
	Virtual citizen science		x		x		x	x	x
	Bitcoin mining	x		x		x			

Table 1.1: **Summary of task computations via worker's attributes**

might also be attracted by the prestige and the reputation gain of having her contribution announced [9]. This last reason for joining the computation together with the fact that a user might actually want to harm the project are enough to jeopardize the reliability of volunteer computing. Several studies [8, 9, 12, 63, 77] have found evidence that reliable task results is not an a-priory property of volunteer computing and there is a need for establishing it.

In addition to users volunteering computational resources, humans themselves connected to the Internet are a source of computational power. The word crowdsourcing was introduced by Howe [67] to describe the situation where human intelligence tasks are executed over the Internet by humans that are given monetary, social or other kind of incentives. A profit-seeking computation platform has been developed by Amazon, called Amazon's Mechanical Turk [102] (AMT). Users sign in to the platform and choose to perform a Human Intelligence Task (HIT) in return for a monetary reward. The most common tasks encountered in AMT are closed class questions (following the categorization in [47]), meaning that the range of answers is a limited predefined set. The word crowdsourcing is vastly used to describe any contribution that can be produced by a human-technology collaboration by having participant volunteer or being paid. In this work we consider micro-task crowdsourcing as it was described above and by convention we refer to it simply as crowdsourcing. Like volunteer computing, crowdsourcing systems can not be considered reliable [47, 68], especially now that participants expect a monetary gain.

Another example of an application that can be considered Internet-based computing is Bitcoin mining [23], that has attracted a huge interest from the users and the financial industry. In Bitcoin mining workers carry out complex computations to validate transactions based on Bitcoins, a digital currency. The computational paradigm is peer-to-peer, that is, there is no centralized authority. Nevertheless, the whole system can be viewed as the master assigning tasks to workers, or in this case, the miners. Given that miners may be deceitful, the system must include measures to prevent or minimize this drawback. However, Bitcoin relies on the complexity of the computation

and Bitcoin payments (proof of work) to guarantee trustworthiness [20].

Finally, another example of Internet-based task computing we could consider is virtual citizen science [41,76]. We could say that this type of computation is a hybrid among what we call volunteer computing and crowdsourcing. Volunteers over the Internet willing to help scientists accept to participate in tasks that need human intelligence to be solved. One of the most characteristic project is Galaxy Zoo [41, 56], engaging volunteers to classify galaxies into categories, doing so in many occasions in the form of a "fun" game. It was through the work of Von Ahn [131], that pioneered *games with a purpose*, where we first saw this type of scheme for creating a more pleasant experience for the volunteers. As pointed out by Kloetzer et al. [76] volunteers will gradually learn to perform a task through these "task-game mechanisms" as they call them. Thus, this type of volunteers actually become reliable over time and given the right incentives.

The Nature of the Workers

All of the aforementioned examples in essence follow a master-worker model. A master process sends tasks, across the Internet, to available worker processes, which execute them and report back the task results. Moreover, it is clear from the nature of Internet-based task computing that workers can not be trusted. A number of works present evidence that workers might actually misreport values in online task computing platforms [8, 9, 47, 63, 68, 75, 77]. The most trivial reason for workers misreporting values is due to hardware or software failures happening during the computation of the task that was not detected. Other explanations are the workers might have ulterior motives for misreporting results or might have limited capabilities of computing a correct result. In order to be able to provide solutions that establish reliability we need first to be able to correctly model the worker's behavior.

A number of attempts have been made in the past to classify these workers. In Distributed Computing a classical approach is to model the malfunctioning (due to a hardware or a software error) or cheating (intentional wrongdoer) as *malicious* Byzantine workers that wish to hamper the computation, and thus always return an incorrect result. The non-faulty workers are viewed as *altruistic* ones [122] that always return the correct result. On the other hand, a game-theoretic approach assumes that workers are *rational* [4, 58, 124], that is, a worker decides whether to truthfully compute and return the correct result or return a bogus result, based on the strategy that best serves her self-interest (increases its benefit).

A common practice in the crowdsourcing literature is to view workers as rationals in the general sense of the term and propose social, financial or hybrid incentives [94, 123] in an attempt to improve on the reported results. Moreover, workers have been considered rationals in a game-theoretic sense and game-theoretic models based on repeated games where proposed and analyzed [134]. Additionally, budget minimizing mechanisms targeting an overall reliability on a number of tasks [73], where considered. Finally, crowdsourcing workers were also viewed as malicious able to harm the computation through a number of attacks and adversarial techniques have been proposed [44] to deal with these attacks.

Given the above, we can safely assume that in an Internet-based task computing system, a number of workers will be reporting always correct task results, a number of workers will be reporting always incorrect results, and a number of workers will be reporting both correct and incorrect results. In order to be able to give solutions to the problem of unreliability of the workers, we need to model their behavior under a general framework. This work looks into two different models of the workers' behavior.

Error probability model: We assume the presence of two type of workers. **Altruistic** workers that are positive towards executing the task, and willing to provide the correct result. In the case of a BOINC system, a "positive" worker will let its machine execute the task and report back the result, while in the case of crowdsourcing she will aim at computing the correct task result. **Troll** workers are negative towards executing a task, and want to convey an incorrect result to the master. Hence, a troll worker can miscalculate a task on purpose by reporting a random incorrect result. Altuistic and troll workers are subject to errors, and thus it is possible that an altruistic worker will report an incorrect result while a troll worker might report a correct result. Notice that in this model we do not assume that workers have an intelligent strategy to fool the system, nor that a troll worker has any information on the actions taken by the master or the other workers.

Rationality model: We assume the presence of three types of workers. **Rational** workers are selfish in a game-theoretic sense and their aim is to maximize their utility (benefit). Thus, they will report a correct or incorrect result based on the strategy that maximizes their benefit. **Altruistic** and **malicious** workers have a predefined behavior, to always be honest and cheat, respectively.

Notice that the concepts of trolls and malicious workers, in the two models mentioned above, are very similar. The main difference among the malicious and the troll workers is that malicious workers have an intention of harming the computation and thus they form even collusions to achieve their goal. On the other hand, troll workers are simply not interested in providing the correct reply.

The Nature of the Masters

Besides workers exhibiting a strategic behavior (aiming at reporting a task result that will maximize their benefit) also the masters profiting from in the computational environment might exhibit strategic behavior. Rationality on the master's side has been also considered in the past [134].

In the case of profit-seeking platforms, such as AMT, a number of requesters (as masters are referred to in AMT) are competing to have their HITs computed. According to a study by Ipeirotis [69] a very small amount of requesters accounts for more than 30% of the overall activity on AMT. That study shows as well that this small percentage of the top contributor requesters

is aiming at workers with the same skills, while their budget varies in the order of thousands of dollars. Thus, competitive behavior might arise among requesters, especially ones that are aiming at popular task types such as content generation or object classification, for example. The way AMT is designed allows requesters having a strategic behavior to increase the reward for a completed task and attract more workers participation. Another feature of AMT is that it gives the possibility to a requester to declare a task result incorrect and refuse payment to the worker. Thus, given the fact that requesters are limited by a budget, they might exhibit a strategic behavior to increase their benefit by posting tasks with high payments and not rewarding the worker after receiving the answer. This strategic behavior does not only affect the workers but also the rest of the requesters competing for workers with the same set of skills. Thus, it is essential in Internet-based task computing to have fair and efficient (when the workers are needed the most) allocation of the workers among the competing masters.

Contributions & Document Organization

We address the issues of reliability and fairness in online task computing environments, proposing solutions intended to alleviate the presence of malicious entities and incentivize the good behavior of strategic entities, in an attempt to exploit the full potentials of this computing environments. In particular, in this work we have three research goals:

Reliability: Achieve a reliable online task computing platform, in the presence of untrustworthy workers, where the master receives with high probability the correct answer.

Worker Characterization: Study the behavior of workers participating in micro-task crowdsourcing and assess the validity of our proposed worker models. In parallel, provide key information to design reliable crowdsourcing platforms.

Fair Distribution: Achieve a fair and efficient distribution of workers among the set of strategic masters. That is, a worker must be executing a task for the master that has the highest valuation for her work, at any time.

To this respect our general contributions can be summarized in the following:

- We provide a comprehensive study on online task computing environments paving the way to a broad adaptation of this easy and inexpensive source of computational power.

- We abstract online task computing environments to the Internet-based task computing system model and we model the computational entities under the master-worker paradigm.

- We model the workers' unreliable behavior under two general models, the error probability model and the rationality model.

- Given the error probability model we consider tasks with multiple correct and multiple incorrect solutions, while a classical assumption is to consider task with a binary set of

solutions, thus the problem complexity increases. For this reason, we provide a study of the two main reliability techniques: (1) auditing and (2) voting given the above assumptions. To do so, we design algorithms using the above techniques that decide with high probability the correct task solution for a set of tasks. We analyze and compare the computational complexity of the algorithms in terms of time and required work from the workers. For a detailed analysis of these contributions we refer the reader to the Contribution paragraph of Chapter 3.

- We model the workers behavior according to the rationality model and we assume a repeated interaction with the workers. The computation is modelled using evolutionary dynamics and we study the conditions under which the master can reliably obtain task results. In particular, we develop and analyze algorithmic mechanisms based on reinforcement learning to provide rational workers with the necessary incentives to eventually become truthful, while we use reputation to deal with malice and unresponsive workers. Our analysis identifies the conditions under which truthful behavior can be ensured. The analysis is complemented with illustrative simulations. Detailed contributions are provided in Chapter 4 in Sections 4.3, 4.4 and 4.5.

- We provide an experimental evaluation of the workers behavior on AMT where we identify workers with different levels of accuracy in their responses. We verify a relationship among reliability and task difficulty and also the relationship among reliability and worker's response time. Most importantly though, we are able to conjecture that validity of our two models characterizing the workers' behavior.

- We design a mechanism for fair and efficient distribution of the workers without the use of monetary incentives in an Internet-based computing system, in the presence of strategic masters aiming at increasing their own benefit by declaring a higher than their actual valuation on a workers' contribution. Analysis shows the good properties of our mechanism while simulations compare the designed mechanism with renown auction mechanisms that use monetary incentives. For more details on these contributions we refer the reader to the Contribution paragraph of Chapter 6.

In the rest of the document we present our work evaluating it analytically, through simulations and experimentally. In Chapter 2 we present the most related concepts to our designed solutions. Additionally, we review the literature regarding reliability and fairness in Internet-based task computing systems. In Chapter 3 we model the behavior of the workers under the error probability model and we review two prominent reliability techniques assuming the presence of tasks with multiple correct and multiple incorrect solutions. We evaluate the performance of the two techniques, in terms of the master receiving the correct task results with high probability, depending on execution time and task assignments. Moving to Chapter 4 we use the rationality model to capture the behavior of the workers. Our design goal is to achieve a reliable computational platform,

in which the master will always be receiving the correct task result after a number of interactions with the workers. We begin by designing solutions that assume only the presence of rational workers, and subsequently we move forward designing solutions that assume the presence of malicious, altruistic and rational workers. Since the master interacts multiple times with the same workers, we design reputation schemes that deal with the malicious workers alleviating their impact in the final task outcome. In Chapter 4 we also present solutions where the master deals with workers being unresponsive. A solution is created where the master selects the workers to assign the task from the whole set of workers, instead of assigning the tasks to a predefined set of workers. Chapter 5 presents the results of an experimental evaluation of the workers' behavior on the AMT platform. The analysis of the received data confirms the presence of workers reporting incorrect task results. Moreover, there is evidence of strategic behavior from the workers. The chapter concludes with a discussion where we argue that based on the experimental data our two models (error probability model and rationality model) capture the workers behavior. Chapter 6 presents a mechanism that solves the problem of fair and efficient distribution of the workers among the masters in the presence of non-monetary incentives. Finally, Chapter 7 summarizes the most important conclusions and provides a discussion on the possible future directions.

Chapter 2

Background and Related Work

There is a great amount of literature related to what we describe in this work as the master-worker Internet-based task computing problem. In this chapter we aim at reviewing the most closely related works giving particular emphasis to how the problem of reliability and fairness is addressed in volunteer computing and crowdsourcing. Moreover, we provide some background on the concepts appearing in this work and we refer the reader to works that extensively describe the concepts of Game Theory (GT), Mechanism Design (MD), Evolutionary Game Theory (EGT), reinforcement learning, reputation and so forth.

2.1 Background

Before reviewing the related work, we briefly introduce some concepts relevant to with this work.

Mechanism Design & Auction Mechanisms

The concept of GT [61, 100, 103] is used in economics, computer science, biology, political science, etc., and tries to capture and study strategic interaction among rational agents driven by their self-interest, beliefs and preferences. In an abstract way we can describe a game a the set of available actions the agents have and a mapping of the agents strategies to an outcome. The utility of an agent, determines her preferences over the possible strategies she can follow and the strategies of the other agents. Generally an agent is considered rational when it tries to maximize its utility. To this respect, solution concepts try to compute the outcome of a game with rational agents. The most famous solution concept is the Nash Equilibrium (NE) [98]. The concept of rationality [61] is used to model the human behavior and formally define the outcome of a game with self-interested agents. A complementary approach to GT is MD [100, 103] where given a targeted outcome, a game is designed to incentivize the rational agents in behaving in such a way such that the desired outcome is achieved.

The concepts of Algorithmic Game Theory [100], MD [103] and Algorithmic Mechanism Design [100] have a large applicability in computer science, in the fields of distributed systems and networks since agents interacting on the Internet are considered strategic [100]. Examples of popular problems addressed through these fields are network routing problems, scheduling problems, task allocation problems, etc,. In MD a designer of a game aims at choosing game structures, called mechanisms that will result in desirable outcomes. In a game we have a system with a number of strategic agents and a possible set of outcomes. Each agent has a type, which is a private information of the agent, determining the agents utility over different outcomes. Thus, an outcome is preferred by an agents if it strictly provides her with a larger utility. The problem addressed is to compute a social choice function that selects an optimal outcome given all the agents types. The most common social choice function is to maximize the total utility of the agents. In order for the mechanism to achieve the desirable outcomes, a set of rules and incentives are designed to influence the rational agents preferences.

The most straightforward mechanisms for resource allocation are auctions [97], where agents submit bids based on their preference on the auctioned good and the outcome is to assign the good to the agent that, for example, maximizes the revenue given the bids. An auction can be categorized in many ways [80]; open or sealed bid, first price auctions or second price auctions. It can be a single object auction or a multi-unit auction. If it is a multi-unit auction, objects can be identical or not and can be sold in a single round or over sequential rounds and objects can be complements (i.e. the value of a second object obtained that is complement of the first one obtained increases). Multi-unit auctions can be classified as uniform price auctions or discriminatory price auctions. In uniform price auctions there is a fix amount of k highest bids that win the auction and winners pay the same amount; bidder also declare the units of good they desire. In the work presented in Chapter 6 we are assuming a multi-unit, sealed, sequential (with many objects auctioned at each round) auction where auctioned goods are not complements. When the agents (bidders) in an auction know how much they value a good, and this information is only known to them, this describes a situation of *private values* [80], which is the case in this work. One of the most celebrated auction mechanisms are the ones belonging to the Vickrey-Clark-Groves (VCG) [16, 100, 103] family of mechanisms, where an agent's dominant strategy is to be truthfully revealing its preferences, independently of the other agents actions. A mechanism that has this property is called strategy-proof.

Evolutionary dynamics & Evolutionary Game Theory

Evolutionary dynamics were first studied in biology as a tool to study the mathematical principles according to which life is evolving [101]. Since then, a number of fields were inspired by the principles of evolutionary dynamics (e.g., sociology, economics, artificial intelligence) and a variety of mechanisms was developed, aiming to accurately model the process of evolution. The work presented in Chapter 4 is inspired by dynamics of evolution as a mean to model workers adaptation to a truthful behavior.

The dynamics of evolution have been studied under the principles of EGT. Maynard-Smith and Price [125, 126] introduced the concept of EGT in an effort to apply game theoretical ideas to understand evolving populations of lifeforms. This made Maynard-Smith [126] adjust the traditional concept of strategy, equilibrium, and the nature of a player's interaction, so that a player would learn how to optimize its behavior and maximize its return. However, while in traditional GT players choose a strategy from their strategy sets, EGT in biology is dealing with species inheriting possibly mutated strategies. When referring to social entities such as humans, evolution is understood as a learning process akin to "cultural evolution" [64]. Cultural evolution implies an analogy between learning and biological evolution. For the existing analogies between learning, at the individual level, and biological evolution, we refer the reader to a paper by Borgers and Sarin [26].

In EGT, instead of the NE [98] , Maynard-Smith and Price used the Evolutionarily Stable Strategy (ESS) concept [45, 57, 115, 132]. A strategy is called evolutionarily stable if, when the whole population is using this strategy, any group of invaders (mutants) using a different strategy will eventually die away over multiple generations (evolutionary rounds). All ESS are NE but the reverse is not true. Our work is driven by the concept of ESS and we wish to have a similar stable strategy among workers that would guarantee reliability. Even if a mutant worker decides to change its strategy to cheating, it will be soon brought back to an honest strategy. Instead of the one-shot and repeated games of classical GT, EGT assumes that the game is played repeatedly by players randomly drawn from large populations, uninformed of the preferences of opponents. In our work we do not wish to change the set of players as evolutionary rounds progress, but we rather talk about "cultural evolution", where workers change their strategies as a process of learning, rather than being replaced themselves.

Reinforcement Learning & Aspiration Level

While evolution operates on the global distribution of strategies within a given population, reinforcement learning [128] operates on the individual level of each member of the population. A well-known model of reinforcement learning is Bush and Mosteller's model [28]. In this model, the players have limited information and they play in discrete time repeatedly the same normal-form game. At each point in time, the players are characterized by a probability distribution over their strategy sets. Players' choices are random, since they are affected by some unpredictable "psychological" factor. This probability distribution is adjusted over time in response to experience. This experience is gained through repeated interactions of the players with the system, based on their strategies and the received payoffs. Positive payoffs reinforce the strategy just chosen, and negative payoffs discourage the use of that strategy.

Specifically, Bush and Mosteller's model is an aspiration-based reinforcement learning model: Players adapt by comparing their experience with an *aspiration* level. There are several models of how aspirations are formed and adjusted over time, formally described in a study by Bendor et al. [21]. In the work presented in Chapter 4, we use a simple model where aspiration is fixed by the

workers and does not change during the evolutionary process (as in [22]). For more information on the different reinforcement learning models and comparisons between them we refer the reader to a paper by Laslier et al. [85] and a study by Izquierdo and Izquierdo [70].

A survey by Phelps et al. [108] and an article by Conitzer and Sandholm [42] take a new approach on MD by introducing the concept of evolutionary mechanism design. Evolutionary mechanism design assumes an engineering approach, based on an incremental process that creates a partly automated mechanism design. The evolutionary mechanism has a continuous interaction and feedback from the current mechanism, as opposed to classical mechanism design, which after the mechanism is introduced in the system, remains in the same NE forever. Looking at it from a different perspective, evolutionary mechanism design is analogous to EGT. Just as players may be forced to gradually adjust their strategies, in an analogous manner mechanisms are gradually making adjustments in their rules with respect to what strategies are currently in play. 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.

Distributed computation in the presence of selfishness was studied within the scope of combinatorial agencies in Economics [17–19, 48]. The basic model considered is a combinatorial variant of the classical principal-agent problem [92]: 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 deals in general 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. One difference with respect to our master-worker model is that, the worker's actions cannot really be viewed as *hidden* in our setting. Another important difference is that our scheme considers worker punishment, as opposed to the schemes in combinatorial agency where workers cannot be fined (limited liability constraint); this is possible in our framework as worker's actions are contractible (either a worker truthfully performs a task or not).

In the work of Rose and Willemain [112] the principal-agent problem is extended to evolutionary learning, and bounded rationality of the agents is assumed. Players' learning is simulated with a genetic algorithm that roughly mimics selection and mutations in biological evolution. Changes in the system are externally induced through the use of incentives. The agents' learning is aided by the principal's incentives that are used to adjust the learning, according to the output the principal desires. The principal is also able to use an artificial selection procedure to identify high performing agents for its own benefit.

Compared with the work of Rose and Willemain [112], in our line of work the learning model is different (in addition to the differences our work has with the principal-agent model). We assume that the learning procedure of the players remains the same through the evolutionary process. In contrast, in the more general model of Rose and Willemain, the learning procedure of

the players may change over time and players can experience mutations. In both works incentives are used to impose a desired behavior over time. But in [112] bounded rationality has been used, while in our work no cognitive limitation of the workers is assumed.

Reputation

In Chapter 4 we use reputation to deal with malice. The master reinforces its strategy as a function of the reputation calculated for each worker. Reputation has been widely considered in on-line communities that deal with untrustworthy entities, such as online auctions (e.g., eBay) or Peer to Peer (P2P) file sharing sites (e.g., BitTorrent); it provides a mean of evaluating the degree of trust of an entity [72]. Reputation measures can be characterized in many ways, for example, as objective or subjective, centralized or decentralized. An objective measure comes from an objective assessment process while a subjective measure comes from the subjective belief that each evaluating entity has. In a centralized reputation scheme a central authority evaluates the entities by calculating the ratings received from each participating entity. In a decentralized system entities share their experience with other entities in a distributed manner. In our work, we use the master as a central authority that objectively calculates the reputation of each worker, based on its interaction with it; this centralized approach is also used by BOINC.

The Berkeley Open-source software for volunteer computing (BOINC) system itself uses a form of reputation [11] for an optional policy called adaptive replication. This policy avoids replication in the event that a job has been sent to a highly reliable worker. The philosophy of this reputation scheme is to be intolerant to cheaters by instantly minimizing their reputation. Our mechanism differs significantly from the one that is used in BOINC. One important difference is that we use auditing to check the validity of the worker's answers while BOINC uses only replication; in this respect, we have a more generic mechanism that also guarantees reliability of the system. Notwithstanding inspired by the way BOINC handles reputation we have designed a BOINC-like reputation type in our mechanism (called Boinc). The adaptive replication policy currently used by BOINC has changed relatively recently. BOINC used to have a different policy [10], where a long time was required for the worker to gain a good reputation but a short time to lose it. In this work we evaluate the two policies used by BOINC, adapted of course to our mechanism. We call the old policy Legacy Boinc and we seek to understand the quantitative and qualitative improvements among the two schemes.

Sonnek et al. [127] use an adaptive reputation-based technique for task scheduling in volunteer setting (i.e., projects running BOINC). Reputation is used as a mechanism to reduce the degree of redundancy while keeping it possible for the master to verify the results by allocating more reliable nodes. In our work we do not focus on scheduling tasks to more reliable workers to increase reliability but rather we design a mechanism that forces the system to evolve to a reliable state. We also demonstrate several tradeoff between reaching a reliable state fast and the master's cost. We have created a reputation function (called reputation Linear) that is analogous to the reputation function used in [127] to evaluate this function's performance in our setting.

2.2 Related Work

In this section we review the most relevant works addressing the issue of reliability in Internet-based task computing systems and fair resource allocation without the use of monetary incentives. Related to the topic of reliability we review a number of works considering the presence of malicious, altruistic and rational workers.

Presence of Malicious & Altruistic Workers

Sarmenta [118] was among the first that introduced the problem on reliability in volunteer computing. In his work he assumed the presence of malicious and altruistic workers, with only the malicious workers having a constant probability of submitting an erroneous result and binary set of response. Based on these assumptions he introduced sabotage-tolerance mechanisms that used voting and a spot checking technique (similar to auditing in our work). Given the assumption that altruistic workers will always reply with the correct task result, the mechanisms combine voting with challenges to create a reputation for each worker and increase the systems' reliability while trying to keep redundancy low.

In the work of Kondo et al. [77], the error detection mechanisms presented in the work of Sarmenta are evaluated with real data gathered from the XtreamLab project that uses the BOINC [8] infrastructure. Through this work the errors generated in desktop grid applications are characterized. They conclude that a large fraction of errors is coming from a very small portion of workers. Also they show that little correlation between simultaneous malicious workers exist. Additionally, there is a large variability of the set of malicious workers over time, with the exception of a few frequent offenders. Also care has to be taken if blacklisting or credibility is used. They derive the conclusion that a large number of tasks and time are required to achieve low error rates with spot-checking and that, in general, to achieve low error rates it is better to use majority voting.

In the work of Fernández et al. [51], an asynchronous distributed system is considered where the master processor sends tasks to a collection of n workers and a worker may deliberately return an incorrect result in an effort to harm the master. Hence, they consider workers to have a predefined behavior: either they are malicious or correct. The authors model with an explicit parameter d the probability that the master will receive a reply from a worker on time. The master employs majority voting in accepting the correct result of a task. In order to analyze a worst case scenario the assumption that malicious workers reply and return the same incorrect result is made (thus a form of collusion is assumed). For each task assigned to a worker the master is charged with one work unit. The goal of the master is to accept the correct task result with high probability of success $1 - \varepsilon$, $\varepsilon \ll 1$, and with the smallest possible amount of work. A probabilistic bound on the number of malicious workers is considered, with a probability $p < 1/2$ of any worker processor being faulty. Lower bounds on the minimum amount of (expected) work required are given, so that any algorithm accepts the correct reply with probability of success $1 - \varepsilon$.

A single communication-round protocol is proposed, where the master decides upon the cor-

rect reply by the end of the round. Two algorithms for the master using different decision strategies are developed: a majority-based and a threshold-based algorithm. The authors show that both algorithms obtain the same probability of success $1 - \varepsilon$ and derive similar upper bounds on the (expected) work required in doing so. Also they show that under certain conditions, these upper bounds are asymptotically optimal with respect to the shown lower bounds. This work shows that it is possible to devise mechanisms for executing tasks reliably with high probability and low cost in the presence of malicious worker processors and unreliable communication, and do so with provable analytical guarantees. The work proposed by Fernández et al. [51] takes into account the conclusions derived by the work of Kondo et al. [77], that in general achieving low errors it is better to use majority. Also considers the unreliability of the network and unavailability of the workers something that is not considered in the work of Sarmenta.

In their work Konwar et al. [78] remove the assumption that the probability of a worker being malicious is known and new algorithms are proposed to approximate this probability. The lower bounds on the amount of work necessary, when the set of responses is binary, are comparable to the work complexity of our proposed algorithms. These works assume that all workers might fail in every round with a certain probability. In our work we take a more realistic assumption, considering that workers might reply incorrectly with a different probability related to their nature (that is, we assume two types of workers' failures).

The work in [81] proposes a distributed verification mechanism in which workers can verify each other's tasks. This approach can potentially aid the master, especially in multi-round computations, from performing the verification by itself. The authors have incorporated their distributed checking mechanism in the BOINC server software. The goal of their mechanism is to give rewards only to altruist workers and prevent the system with being flooded with incorrect results. Here the malicious workers are assumed to form a single coalition. The focus in our work is not to design a verification mechanism, but instead to limit the use of auditing to the minimum (in such a case, the master does not suffer a prohibitively large overhead for auditing). Also, the verification mechanism developed in [81] assumes that no more than 20% of the workers are malicious; otherwise their mechanism fails to verify the correctness of a result. This bound is significantly smaller than the one assumed in [51] (50%). Our work, is not subject to any such limitations (due to the central auditing employed by the master), but it does trade cost with reliability (more auditing is needed if the number of malicious workers is large).

In their work Zhao and Lo [135] compare voting to challenges (called Quiz) under two assumptions: that all malicious workers return the same incorrect result or that malicious workers return distinct results. They use as performance metrics the accuracy and overhead, and through simulations they show the trade-off among these performance metrics and the two reliability techniques. Their work is mostly experimental and does not provide any complexity analysis. Additionally they do not assume a density of solutions nor an error probability on the altruistic workers.

Karger et al. [73] propose an algorithm that minimizes the degree of redundancy needed

for each task while achieves an aimed total reliability of the tasks on a one-shot assignment. Their proposed algorithm assigns tasks to workers based on a bipartite graph and based on that graph their inference algorithm decides upon the correct task result of the task after a number of iterations. They conclude that if they have a certain information on the quality of the crowd their algorithm performs better than using majority to infer the correct solution. Like in our work a kind of malicious and altruistic workers are assumed, named spammers and hammers. Unlike in our work here spammers will submit a correct answer with probability 1/2 and hammers will always submit a correct answer. More over tasks with binary solutions are assumed. Like us thought they assume that tasks have the same difficulty and the probability of replying incorrectly does not depend on the particular task.

Finally the work of Laredo et al. [84] takes a different approach designing scalable evolutionary algorithms resilient to failures, that is to errors arriving from potentially malicious workers, and analyzes experimentally the speed by which the system converges. In their experiments they considered two paradigms of evolutionary computation: genetic algorithms and genetic programming. Their model assumes the presence of workers that join and leave the computation for a limited period of time, an assumption that we explore as well in Section 4.5.

Presence of Rational Workers

Under the classical game-theoretic view, workers act on their own self-interest and they do not have an a priori established behavior (malicious or altruistic). They are assumed to be rational [4, 58]. In other words, the workers decide on whether they will be honest and report the correct task result, or cheat and report a bogus result, depending on which strategy increases their benefit or utility. Under this view, Algorithmic Mechanisms [4, 99] are employed, where games are designed to provide the necessary incentives so that processors' interests are best served by acting "correctly." In particular, the master provides some reward (resp. penalty) should a worker be honest (resp. cheat). The design objective is for the master to force a desired unique NE [98], i.e., a strategy choice by each worker such that none of them has an incentive to change it. That NE is the one in which the master achieves a desired probability of obtaining the correct task result.

The work by Yurkewych et al. [133] considers computational grids where clients are financially compensated for their work. The work considers rational workers [124], that is, the workers do not have a pre-defined behavior, but instead they are selfish and seek to maximize their expected profit. In particular, the workers follow a cheating strategy only if that increases their expected profit compared to an honest strategy. As with the work in [51], redundant task allocation is employed: the master sends the same task to several workers and collects their replies. Also it is assumed that workers collide into teams, and each team returns the same result. Since workers are rational in a game-theoretic sense, incentives are given to the workers in order to be truthful. If workers are caught cheating, they are penalized. For this purpose, an auditing mechanism is used. This work assumes that the communication is reliable. In more detail, in [133], a game is developed, called Internet Auditing Game as follows: The master chooses a set of work-

ers, announces the probability by which it will audit and sends the task to the workers. If the master audits then it rewards honest workers and penalizes cheaters. If it does not audit and there is a strong majority of workers (more than 50%) that return the same result, then it rewards the members of the majority and penalizes the rest. If there is no strong majority, then no reward/punishment takes place, the master chooses again randomly a set of workers and re-initiates the same process with the same task (but no knowledge is used from the previous round). That is, not always a single-round protocol is followed for a specific task, whereas our frameworks consider a single-round interaction per task. As indicated experimentally by Kondo et al. [77], a task may take more than one day of Central Processing Unit (CPU) time to complete. Hence, repeating the computation of a task seems to waste useful computation time.

The master in [133] has a fixed budget for computing a task, that includes the auditing cost and the rewards to the workers cost. The goal of the master is to guarantee that it will get the correct reply without exceeding its budget. Bounds for the auditing probability are computed to guarantee that workers have incentives to be honest in three scenarios: redundant allocation with and without collusion, and single-worker allocation. The authors conclude that single-worker allocation is a cost-effective mechanism, especially in the presence of collusions. In our work we do not restrict the budget of the master, but instead, we show trade-offs between reliability and cost (hence considering budgets implicitly).

A later work by Fernández et al. [52] considers Internet-based master-worker computations from a game-theoretic point of view. These computations are modeled as games where each worker is assumed to be rational and can choose to cheat (i.e., fabricate a reply and return it to the master), or be honest (i.e., compute and return the correct result). A general single-round protocol is presented where the master assigns the task to n workers. Each worker cheats with a probability and the master verifies the answer with some probability. If the master verifies, then it rewards and punishes workers appropriately. If the master does not verify, then it rewards the workers according to one of three reward models: (a) "majority" reward model, where the majority of workers is rewarded, (b) "all" reward model the master rewards all workers and (c) "none" reward model the master does not reward at all. Cost-sensitive mechanisms that provide the necessary incentives for the workers to truthfully compute and return the correct result are designed. The objective is to maximize the probability that the master will obtain the correct task result while minimizing its cost. For this purpose the authors consider a set of realistic payoff parameters that can model the environment considered in a game-theoretic sense. Four different games are considered: (a) a game between the master and a single worker ($1 : 1$ game), (b) a game between the master and a workers played n times, each with a different worker ($1 : 1^n$ game), (c) a game with a master and n workers ($1 : n$ game) and finally, (d) a game with n worker and the master participating indirectly ($0 : n$ game). Combined with the three reward models the authors have considered twelve games in total. The authors analyze the conditions under which a unique NE [98] is reached for each of the twelve games. Thus, the analysis leads to mechanisms where the master can choose the game conditions that guarantee a unique NE that best fits its

goal. Finally they have identified and proposed specific mechanisms for two realistic scenarios, a volunteer computing scenario (like Search for Extraterrestrial Intelligence (SETI)@home [79]) and a profit-seeking company that buys computational cycles from Internet computers and sells them to customers in the form of a task-computation service (like Amazon Mechanical Turk (AMT) [102]).

The authors in [52] consider a weak form of collusion, where all cheaters return the same incorrect result (we make the same assumption in our work as well). This leads to a worst case analysis, covering all other scenarios with weaker types or no worker collusions. Also they make the assumption that if a worker does not perform the task, then it is almost impossible to guess the correct answer (in other words, guessing the correct result is very unlikely); we make this assumption in our work as well (see Chapter 4). Also, in [52] a verification mechanism is assumed where the master can verify which workers have truthfully performed the task, without computing the task. Hence, the master by verifying does not necessarily obtain the correct result, for example when all workers cheat.

Compared with the work of Yurkewych et al., the work of Fernández et al. [52] studies more algorithms and games, considers richer payoffs, probabilistic cheating and shows reacher trade-offs between reliability and cost. However they share a common conclusion: under certain conditions non-redundant task allocation is best. This motivates the need for better mechanisms that would make redundant allocation more effective. Considering knowledge gained in previous interactions for future task computations is a promising direction, as we observe in this work.

Also in the work of Fernández et al. [53] workers are considered rational behavior on the side of the workers in a game-theoretic sense. The authors considered that rational workers might collude and they analytically present parameter conditions where a unique NE exist and the master obtains the correct answer. Finally, in [134] the authors consider a crowdsourcing setting and present a game-theoretic model based on repeated games, where workers and masters are considered selfish. They propose a class of incentive protocols that are based on social norms and analyze the social welfare of the platform proving that it can operate close to Pareto efficiency.

Presence of Altruistic, Malicious & Rational Workers

Eliaz [49] seems to be the first to formally study the co-existence of Byzantine (malicious) and rational players. He introduces the notion of k*-fault-tolerant NE* as a state in which no player benefits from unilaterally deviating despite up to k players acting maliciously. He demonstrates this concept by designing simple mechanisms that implement the constrained Walrasian function and a choice rule for the efficient allocation of an indivisible good (e.g., in auctions). Abraham et al [4] extend Eliaz's concept to accommodate colluding rational players. In particular they design a secret sharing protocol and prove that it is (k, t)-robust, that is, it is correct despite up to k colluding rational players and t Byzantine ones.

Aiyer et al. [5] introduce 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. (In this respect, one might say that our aim is to design mechanisms that are BAR-tolerant.) As an application, they designed a cooperative backup service for P2P systems, based on a BAR-tolerant replicated state machine. Li et al [90] also considered the BAR model to design a P2P live streaming application based on a BAR-tolerant gossip protocol. Both works employ incentive-based game theoretic techniques (to remove the selfish behavior), but the emphasis is on building a reasonably practical system (hence, formal analysis is traded for practicality). Moreover, Li et al [89] developed a P2P streaming application, called FlightPath, that provides a highly reliable data stream to a dynamic set of peers. FlightPath, as opposed to the above-mentioned BAR-based works, is based on mechanisms for *approximate equilibria* [30], rather than strict equilibria. In particular, ϵ-Nash equilibria are considered, in which rational players deviate if and only if they expect to benefit by more than a factor of ϵ. As the authors claim, the less restrictive nature of these equilibria enables the design of incentives to limit selfish behavior rigorously, while it provides sufficient flexibility to build practical systems.

Gairing [55] introduced and studied *malicious Bayesian congestion games*. These games extend congestion games [113] by allowing players to act in a malicious way. In particular, each player can either be rational or, with a certain probability, be malicious (with the sole goal of disturbing the other players). As in our work, players are not aware of each other's type, and this uncertainty is described by a probability distribution. Among other results, Gairing shows that, unlike congestion games, these games do not in general possess a NE in pure strategies. Also he studies the impact of malicious types on the social cost (the overall performance of the system) by measuring the so-called *Price of Malice*. This measure was first introduced by Moscibroda et al [96] to measure the influence of malicious behavior for a virus inoculation game involving both rational (selfish) and malicious nodes.

In an article by Alon et al. [7] the notion of Bayesian ignorance is presented. Bayesian ignorance is quantified by comparing the social cost obtained by players that have local views in a Bayesian game to the expected social cost of players with global views. The authors assume the existence of both altruistic and rational players and present their derived results on a specific congestion game. The main result reached is that having rational agents bear a local view is best for the social cost. Relating to our model all workers and the master have the same view of the system, having workers with different views is something that can not be applied to our framework.

Besides investigating the co-existence of malicious and rational players, also the co-existence of altruistic and rational players has been considered. Hoefer and Skopalik [65] study congestion games with altruist players, assuming a level of altruism β_i for each player i: $\beta_i = 0$ being a pure selfish and $\beta_i = 1$ being a pure altruist player. The work of Kuznetsov and Schmid [83] describes arbitrary social relationships between players through a social range matrix. Their work considers the existence of different degrees of rationality or altruism, and the existence of malicious players as well. Their definition of maliciousness and altruism is with respect to the whole set of players.

(I.e., malicious players aim at reducing the utility of the rest of players, while altruistic players aim at increasing these utilities.) Instead, in the context of master-worker task computing, we assume that maliciousness and altruism is with respect to the master.

In an earlier work [31], we considered the presence of malicious, altruistic and rational workers in the presence of an unreliable communication mean and unavailable workers. Two algorithmic mechanisms were designed providing incentives to the rational workers to act correctly, while alleviating the malicious workers' actions and coping with the unreliability of the network. The designed solutions considered a one-shot (or stateless) interaction of the master with the workers, in the sense that the master decides about the outcome of an interaction with the workers involving a specific task, without using any knowledge gained by prior interactions. Each of the designed mechanisms implements an instance of a Bayesian game. Assuming that the stochastic distribution of the workers over the three types is known, analysis provides the probability of the master obtaining the correct result and the master's utility, while identifies the conditions under which the game has a unique NE.

Resource Allocation & Mechanisms without Monetary Incentives

The concept of mechanisms without money has been studied before by Schummer and Vohra [120] and Procaccia and Tennenholtz [110]. We are focusing on mechanisms for resource allocation/distribution. To this respect, a related work is the one by Guo et al. [60]. They study the problem of allocating a single item over multiple competing agents in a repeated setting, without the involvement of money. To do this, they introduce an artificial payment system; they propose a number of repeated Bayes-Nash incentive compatible mechanisms and analyze how competitive they are with mechanisms using money. In their setting they assume, like we do, that agents learn their private values before each interaction, and also that preferences are i.i.d according to a distribution that does not change over time. In a later work, Guo and Conitzer [59] design a strategy proof mechanism for allocating multiple heterogeneous goods among two agents, in a single shot, prior-free and payments-free setting. Our designed mechanism which we call Fair and Efficient Distribution of Resources (FEDoR) promotes a truthful declaration of values, since telling the truth increases the agents benefit (assuming the possibility of the agents being more than two) independently of the other agents strategy. Our mechanism considers a setting of multi-round interactions among the agents, for allocating a set of heterogeneous goods to them. We assume that all agents distributions can be transformed into a uniform [0,1] distribution, but besides that we have no information about the distribution of the agents. So, FEDoR is payment-free mechanism that achieves fairness and efficiency.

The work by Moscibroda and Schmid [95] investigates mechanisms without payments for throughput maximization and compares their social welfare with payment mechanisms. In comparison with our work, these mechanisms can be applied in a non-repeated setting, but it is not guaranteed that a feasible solution in terms of social welfare can be found. Moscibroda and Schmid shed light on the degree up to which payments are inevitable and the potential benefit from

the use of mechanism without payments. In contrast to the kind of mechanisms they consider, in FEDoR there is no need of a trusted entity. Due to the infeasibility of using money incentives in many computer science oriented problems, new techniques developed to substitute monetary incentives, such as the *tit-for-tat* in BitTorrent [91,109] and money-burning mechanisms [62]. Unfortunately, these techniques can not be easily generalized to other problems. In particular, in [88] interdomain routing is analyzed from a game theoretic perspective and incentive-compatible distributed mechanisms without payments are designed in a repeated setting.

Rahman et al. [111] considers the fairness constraint focusing on P2P systems, and propose an alternative approach to resource allocation that achieves fairness and efficiency on effort-based incentives, as opposed to contribution or output-based incentives. This work is somehow related to ours by the fact that we also consider that fairness is not proportional to the valuations of the agents, but in our work no incentives are necessary to achieve efficiency. In addition, we give analytical proofs of the properties of our mechanism, unlike the work by Rahman et al. [111].

Jackson and Sonnenschein [71] presented the concept of linking mechanisms. They showed that when a lot of independent copies of the same decision problem are linked together, then no incentive constrains are needed for agents to be truthful. The spectrum of players' responses to a probability distribution is known by considering a budget restriction. They showed that a linking mechanism is valid when the players' possible decisions are distributed following discrete probabilities. FEDoR is inspired by the work of Santos et al. [117], where they are faced with an orthogonal problem, assigning to one of the participants, same in each round, the execution of a single task that is everyone wishes to see accomplished but nobody wants to execute. The designed mechanism is called Quid Pro Quo (QPQ) and also uses the concept of linking mechanism. It provides a fair way of optimal assignments (total cost is minimized), without using any payment. In our work we wish to achieve the opposite, we have a set of goods (i.e., workers) that we want to be fairly allocated among the agents (i.e., masters). Like in the work of Santos et al. [117] we do assume a linking mechanism and the presence of a GoF (Goodness of Fit test) that decides whether the declared values of a players follow a uniform distribution. Two fundamental deferences among our FEDoR and QPQ is: (1) they are solving problems with opposite goals, (2) our mechanism considers multiple good assignment in a single round while QPQ solves the problem of single task assignments.

Chapter 3

Evaluation of Reliability Techniques

3.1 Introduction

We consider a distributed system that carries out computational tasks, as we have seen in the introductory section. A master entity has a set of computational tasks to resolve that is unable or unwilling to compute locally. Hence, she assigns these tasks over the Internet to worker entities willing to perform the task and reply back to the master with a result. The inherent limitation of this load distribution scheme is the unreliable nature of the workers. We assume that there is no mean of verifying an answer provided by a worker unless we know the set of solutions for the particular task.

Evidence exist that workers might actually misreport values [8, 9, 47, 63, 68, 77]. The most straight forward explanation is that workers might have ulterior motives for misreporting a result. Another reason might be that actually a hardware or software failures happened during the computation of the task that was not detected. Another possibility is, in the crowdsourcing example, that workers intend to convey a correct result but they miscalculate.

Drawing from the above examples and the work of Kondo et al. [77], where they have characterized errors in Berkeley Open-source software for volunteer computing (BOINC) systems, we can infer the existence of two type of workers and we characterize them based on what we call the *error probability model*, where we have **(i) altruistic**[1] **workers:** This type of worker is positive towards executing the task, and willing to provide the correct result. In the case of volunteer computing systems, a "positive" worker will let its machine execute the task and report back the result. While in the case of crowsourcing, workers will intend to compute the correct result. **(ii) Troll**[2] **workers:** This type of worker is negative towards executing a task and wants to convey an incorrect result to the master. Hence, it can miscalculate a task on purpose and tries to report an incorrect result. In this work we do not assume any type of intelligent strategy to fool the system,

[1] For historical reasons, to match with original work in the field of master-worker task computing.

[2] Historically they are called malicious workers, but since we are not assuming any intelligent behavior to harm the system here we call them troll workers. For more information see Chapter 2.

nor that a troll has any information on the actions taken by the master or the other workers.

We can safely assume that both types of workers can be susceptible to a small error probability ϵ that is related to hardware or software failures or limited capabilities (in the case of humans computing the task) or to any other factor that would force them to deviate from their intended behavior. Thus an altruistic worker, will have a high probability of reporting a correct result, while a troll will have a low probability of reporting a correct result. Although this work was initially motivated by the volunteer computing example based on the existing literature, our experimental work, presented in Chapter 5, shows that even in a crowdsourcing setting where humans act in a more "selfish" way, the above modelling can also capture the crowdsourcing setting. Altruistic workers, will aim at reporting a correct task result, by trying to calculate a correct result to the extend of their cognitive capabilities. On the hand trolls, want to provide an incorrect solution, by reporting an apparently incorrect task result without verifying if the reported result is correct or not.

The goal of the master is to identify the correct result for each task assigned with high probability (whp). Two principal techniques are used (individually or combined), by the literature, for increasing reliability. **(i) Voting:** The master uses redundancy by assigning the same task to multiple workers. After all the task replies are collected, the master uses a voting scheme [24, 82, 106] to decide on the correct result. This may fail to provide optimal reliability since a high concentration of incorrect results reported may lead to a decision on the task result that is incorrect. **(ii) Auditing:** The master uses a set of tasks with known solutions, called challenges, to identify the workers that are replying correctly. Again this approach can not guarantee optimal reliability if the same workers in different time intervals provides correct and incorrect results due to an error during the computation of the task (as we discussed above). Similar or identical concepts to this approach are encountered by the name of spot-checking, challenges or quiz [118, 135].

Both techniques add an extra load on the computation of the task. On the one hand, with voting the same task needs to be executed by multiple workers, thus the resources of the system are not used in an optimal way. On the other hand, auditing requires that the workers compute tasks with known solutions, thus not only resources are not used in an optimal way but also the execution time of a task increases. This is a worst case auditing assumption, since the master does not have the power to compute by itself all the possible solutions of a task and has to resolve to task with already known solutions during auditing.

An added difficulty when using these techniques is related to the nature of the task. If you have tasks that can have multiple correct and multiple incorrect results, maybe you can not guarantee that the result of each task will be correct whp. The common assumption in the literature is that tasks have a binary set of solutions (i.e. only one correct and one incorrect solution are feasible).

Voting and auditing have been widely used either alone or in combination to provide the correct task result for each task whp. What is yet unclear is the advantage of one technique over the other or the combination of these techniques in terms of time and work complexity, given

tasks with multiple correct and multiple incorrect results.

Contributions

Our contributions can be summarized as follows:

- We model the master-worker paradigm in the presence of altruistic and troll workers using five parameters $\langle \epsilon, f_a, s, r, T \rangle$ (see Section 3.2): (i) ϵ is the probability that an altruistic worker may reply with an incorrect result or a troll worker with a correct result, (ii) f_a is the fraction of altruistic workers over the set of workers, (iii) s, r are the number of correct and incorrect answers for a task respectively, leading to the more realistic assumption that tasks are not only binary but rather may have solutions in a broader domain, and (iv) $T \subseteq \{C, V\}$ the set of reported result evaluation techniques, i.e., *auditing* using *challenges* and *voting*. Additionally we define two measures: (i) "time" and (ii) "work", for evaluating the complexity of the proposed algorithms.

- We fix $\epsilon = 0$ in Section 3.3, i.e., the simple case where altruistic workers always reply with a correct result and troll workers always with an incorrect result. Given this idealistic scenario, we identify asymptotically optimal bounds on the time and work complexity when auditing and voting are used separately. While it is clear that when auditing is used the master can receive the correct task result with probability one, this is not always the case when voting is used alone. In the case of voting, we were able to show a negative result giving conditions on the parameters of s, r, n_a and n_t, where the master will not always be deciding on the correct task result with probability one. This result reveals that the domain of reported results is important even in the simple case where $\epsilon = 0$.

- We then make the realistic assumption that $\epsilon > 0$ (Section 3.4) and we provide two algorithms MWMIX and MWVOTE that solve correctly all the tasks whp. Both algorithms assume that s, ϵ and f_a are known. Algorithm MWMIX uses both auditing and voting, and can be applied if $1 - \epsilon > \frac{s}{s+1}$. Algorithm MWVOTE uses only voting, and can be applied if $f_a(1-\epsilon) + (1-f_a)\epsilon > \frac{s}{s+1}$. What is interesting to observe is that these algorithms have a log factor overhead compared to the case where $\epsilon = 0$, which as shown in [78], is a necessary price to pay when voting is used.

- Finally, in the case where ϵ and f_a are not known, in Section 3.5 we provide algorithm E_1 that estimates these parameters within tight bounds *whp*.

3.2 Model

Our setting consists of a master process M and a set $W = \{w_1, \ldots, w_n\}$ of n worker processes. Workers might be unreliable and produce an incorrect result.

Definition 3.1 (Problem statement.)**.** *The master must guarantee whp (see below) the correct result for each task t_i where $t_i \in \mathcal{T} = \{t_1 \cdots t_n\}$, without computing the task locally.*

To keep the pseudocode simple for the purpose of exposition we assume that $|\mathcal{T}| = n$. (The algorithms presented in this work can be easily extended to run for $|\mathcal{T}| = O(poly(\log n))$ without violating the statements of correctness.)

Computation & Communication model

Processes in the system communicate by exchanging messages via reliable communication channels. Computation proceeds in synchronous rounds. For a process p a round consists of the following steps: (i) receive incoming messages, (ii) perform computation on the received messages and produce a set of outgoing messages, and (iii) send the produced messages. During a round each worker can compute only one task from the master, and report back the result of the task. A synchronous algorithm L is a collection of processes, and its state is defined over the vector of the states of each process in that collection. An execution ξ of L is an (infinite) sequence of states. We denote by $execs(L)$ the set of executions of L.

Performance Measures

An algorithm L is evaluated in terms of: (i) *time*, and (ii) *work*. Time is defined as the number of rounds that an algorithm L requires in order to determine the result of all n tasks in $\mathcal{T}$. Work represents the number of aggregated results computed by each worker in algorithm L. More formally, let $comp_\xi(w, t)$ be the number of times a worker w computes task t during an execution ξ of algorithm L. The task t can be one the tasks in $\mathcal{T}$ or a task chosen from a set $\mathcal{C}$ of challenge tasks available to the master M. Then,

$$work_\xi = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} comp_\xi(w_i, t_j) + \sum_{t \in \mathcal{C}} comp_\xi(w_i, t) \right)$$

is the work of all the workers in ξ. Thus, the work of an algorithm L is defined as $Work(L) = \max_{\xi \in execs(L)}(work_\xi)$ over all executions of L. The work captures the redundancy used by an algorithm (i.e., the number of workers that compute the same task), as well as the computation performed on auditing.

Density of Solutions

A reported result v for a task t may take values from a domain $D(t)$. Let $S(t) \subset D(t)$ be the set of *correct solutions* and $R(t) = D(t) \backslash S(t)$, the set of reported results that are incorrect solutions for task t. We only consider tasks t that have at least one correct solution, i.e., $|S(t)| \geq 1$.

Given $D(t)$ and $S(t)$ we define the *density of solutions* for task t as $\frac{|S(t)|}{|D(t)|}$. The density of solutions affects the techniques used to determine the correct task result. For simplicity of presentation, we will assume that the size of the domain $d = |D(t)|$, the number of correct

solutions $s = |S(t)|$, and the number of incorrect solutions $r = |R(t)|$ are the same for every $t \in \mathcal{T}$. Hence, the density of solutions is the same for all tasks t.

Worker Types

We assume that each worker $w_i \in W$ is either *altruistic* or a *troll* and its type remains the same throughout the execution of the algorithm. Let $W_a \subseteq W$ be the set of altruistic and $W_t \subseteq W$ the set of trolls. We use $n_a = |W_a|$ and $n_t = |W_t|$ to denote the sizes of these sets. We assume that $n_a \geq 1$. Observe that $W = W_a \cup W_t$ and $n = n_a + n_t$. We also use $f_a = \frac{n_a}{n}$, to denote the fraction of altruistic workers. Altruistic workers are "positive" towards executing a task, always aiming at returning a correct result. Trolls are "negative" towards the task, always trying to provide an answer from the set of incorrect solutions. All workers are subject to an *error probability* ϵ that deviates a worker from its specification. For simplicity of presentation we will assume that ϵ is the same for all workers. In particular, an altruistic worker replies with a correct result with probability $1 - \epsilon$ and with an incorrect result with probability ϵ, while for a troll holds the contrary. For all workers, the correct and incorrect results are selected uniformly at random from the respective sets $S(t)$ and $R(t)$ for a task t. When $\epsilon = 0$ an altruistic worker always replies with the correct result and a troll always replies with an incorrect result.

Result Evaluation

We assume that the master can use two techniques to determine the correct task result: *auditing* (C) and *voting* (V) . During auditing a *challenge* task is used, that is a task which the master knows the correct result. When a challenge is used, the master knows if a worker replied with a correct or an incorrect result. This information can help the master determine the correct result for each task in $\mathcal{T}$. Notice, that here we do not assume that during auditing the master can compute the task itself and determine all the correct task results. When *voting* is used, the same task is assigned to multiple workers, and the master uses some voting technique to decide upon the correct task result.

Definition 3.2 (Environmental Parameters.)**.** *A master-worker system environment can be characterized by the parameters (ϵ, s, r, f_a, T) where: (i) ϵ is the worker error probability, (ii) s is the set of correct replies for each task t, (iii) r is the set of incorrect replies for each task t (iv) f_a the fraction of altruistic workers, and (v) $T \subseteq \{C, V\}$ the technique used for the evaluation of the reported results.*

Probabilities

We use the common definition of *an event $\mathcal{E}$ occurring whp* to mean that $\mathsf{Pr}[\mathcal{E}] = 1 - O(n^{-\alpha})$ for some constant $\alpha > 0$.

3.3 Exact Worker Behavior ($\epsilon = 0$)

In this section we examine the simple case where it is given that the error probability of each worker is $\epsilon = 0$. Hence, altruistic workers always reply with a correct result and trolls always reply with an incorrect result. This scenario is somewhat idealistic in practical master-worker systems, but it is used here to provide the best case analysis of our problem. The results of this section will be used as a reference from the next section, in order to evaluate the performance of the proposed algorithms compared to this optimistic scenario. Notice that any algorithm requires at least $\frac{n}{n_a}$ rounds to complete the computation of all the $|\mathcal{T}| = n$ tasks, and needs to perform n work, if all tasks have to be computed correctly with full reliability.

Algorithm 1 Simple algorithm MWSIMPLE_0 where $\epsilon = 0$ and $T = \{C\}$.

1: Send challenge task t to all workers in W
2: $R[j] \leftarrow$ result received from $w_j \in W$, $j \in [1, |W|]$
3: $U_a \leftarrow \{w_i | R[i]$ is correct$\}$
4: **for** $i = 1 : |U_a| : n$ **do** ▷ for loop increments i by $|U_a|$
5: Send task t_{i+k-1} to kth worker in U_a, $k \in [1, |U_a|]$
6: Add received result for t_{i+k-1} into $Results[i + k]$
7: **end for**
8: **return** $Results$

In the simple algorithm MWSIMPLE_0 that appears in Algorithm 1, the set of altruistic workers is not known and thus we use auditing ($T = \{C\}$) to determine it in a single round. Once the altruistic workers are identified the master makes unique task assignments to each of them. Hence the master needs $1 + \frac{n}{n_a}$ rounds to decide for n tasks, and requires $2n$ amount of work.

Theorem 3.1. *Algorithm* MWSIMPLE_0 *has asymptotically optimal time* $\Theta(\frac{n}{n_a})$ *and optimal work* $\Theta(n)$*, and compute all the* n *tasks with probability 1, when* $\epsilon = 0$*.*

Looking more closely at the general case of algorithms that only use voting ($T = \{V\}$) we have found that it is possible to solve all the tasks efficiently if $n_a > s \cdot n_t$. The algorithm MWVOTE_0 that solves the problem is given in Algorithm 2. In this algorithm the master sends the first task t_1 to all the workers. No incorrect returned value can appear more than n_t times, while from the pigeonhole principle at least one correct value appears at least $n_a/s > n_t$ times. Then, the workers that return values with multiplicity larger than n_t are all altruistic. These workers are stored in U_a and used to solve the rest of tasks. The size of U_a is at least $n_a - n_t(s - 1) > n_t$, and hence the master needs at most $1 + \frac{n-1}{n_a - n_t(s-1)}$ rounds and $2n - 1$ work.

From the above we have the following theorem:

Theorem 3.2. *The algorithm* MWVOTE_0 *compute all the* n *tasks with probability 1 when* $\epsilon = 0$ *and* $n_a > s \cdot n_t$*. It has time* $O(\frac{n}{n_t})$ *and optimal work* $\Theta(n)$*.*

In the case that $n_a = s \cdot n_t$ we have a negative result.

Algorithm 2 Simple algorithm MWVOTE_0 where $\epsilon = 0$, $n_a > s \cdot n_t$, and $T = \{V\}$.

1: Send task t_1 to all workers in W
2: Add worker w_j to set $R[v]$ if it replied with value v
3: $U_a \leftarrow \bigcup_{v:|R[v]|>n_t} R[v]$
4: $Results[1] \leftarrow$ any value $v : |R[v]| > n_t$
5: **for** $i = 2 : |U_a| : n$ **do** ▷ loop increments i by $|U_a|$
6: Send task t_{i+k-1} to kth worker in U_a, $k \in [1, |U_a|]$
7: Add received result for t_{i+k-1} into $Results[i + k - 1]$
8: **end for**
9: **return** $Results$

Theorem 3.3. *If $\epsilon = 0$ and $n_a = s \cdot n_t$, then for any $r > 0$ there exists no algorithm that allows the master node to returns the correct result of a task t with probability greater than $\frac{s}{s+1}$ in any execution.*

Proof: Lets assume that there exists such an algorithm L_m. The master M assigns the task t to a subset W' of the set of workers W. We assume w.l.o.g. that it sends the task to all the workers in W, since L_m can disregard the replies from the workers not in W'. Note that the master does not have any prior information on the correct answers or the answers that each worker returns. So we can examine possible executions for the same task t, where the incorrect results and the troll workers are different. Since $\epsilon = 0$, an altruistic worker always returns a correct answer and a troll returns an incorrect answer. We assume that the same worker returns the same answer if asked to compute the task more than once.

Consider now an execution ξ_1 of L_m constructed as follows. Let $D(t) = \{a_1, \ldots, a_s, a_{s+1}, \ldots, a_{s+r}\}$ be the domain of possible replies for t, where $S(t) = \{a_1, \ldots, a_s\}$ is the set of correct results for t and the rest of the answers are incorrect. Since $\epsilon = 0$ then the master receives n_a correct answers and n_t incorrect answers. Let us assume, that the troll workers are the set $\{w_{n_a+1}, \ldots, w_{n_a+n_t}\}$. Furthermore in ξ_1, let each answer $a_i \in S(t)$ be returned by $\frac{n_a}{s}$ workers. W.l.o.g assume that workers $\{w_{(i-1)\frac{n_a}{s}+1}, \ldots, w_{i\frac{n_a}{s}}\}$ reply with answer $a_i \in S(t)$. Observe that all the trolls reply with the same incorrect answer a_{s+1}. Since, $n_a = s \cdot n_t$, it follows that $n_t = \frac{n_a}{s}$ troll workers reply with a_{s+1}. Since, according to our assumption, algorithm L_m returns a correct answer with probability $p_c > \frac{s}{s+1}$, then it follows by the pigeonhole principle, that L_m will return some answer $a_i \in S(t)$ with probability $Pr[a_i] > \frac{s}{(s+1)s} = \frac{1}{s+1}$, in ξ_1. Let w.l.o.g. a_1 be that answer.

Assume now a second execution ξ_2 which is similar with ξ_1 with the difference that the troll workers are the set $\{w_1, \ldots, w_{\frac{n_a}{s}}\}$ and the correct answers is the set $S(t) = \{a_2, \ldots, a_{s+1}\}$. Each answer is returned by the same set of workers as in ξ_1. Note that correct workers $\{w_{n_a+1}, \ldots, w_{n_a+n_t}\}$ all reply with a_{s+1}. Also the troll workers in ξ_2 all reply with a_1. Since all the workers reply with the same answers to the master in both ξ_1 and ξ_2, and since the master does not have any prior info on the correct answers, then M will not be able to distinguish ξ_1 from ξ_2. Thus, if according to L_m, M returned a_1 with probability $Pr[a_1] > \frac{1}{s+1}$ in ξ_1 then M will return a_1 with the same probability in ξ_2 as well. Since a_1 is an incorrect answer in ξ_2, then M

returns a correct answer with probability $\mathbf{Pr}[\text{ return } a_i \in S(t)] = 1 - Pr[a_1] < 1 - \frac{1}{s+1} = \frac{s}{s+1}$. This however contradicts our initial assumption that L_m returns a correct answer with probability greater than $\frac{s}{s+1}$ in ξ_2 and completes our proof. ■

Looking at the above two results one may conjecture that $n_a = s \cdot n_t$ is in fact the boundary between solvability and unsolvability. However, this is not the case, since, for instance, if $r = 1$ and n_a is not a multiple of n_t, even if $n_a < s \cdot n_t$ it is possible to solve all tasks efficiently. Using the opposite logic the incorrect value will appear n_t times, while from the pigeonhole principle at worst one correct value appears $n_a \mod n_t$ times. Then, the workers that return values with cardinality smaller than n_t are all altruistic. These workers like in Algorithm 2 can be stored in U_a and used to solve the rest of tasks following an analogous algorithm. The size of U_a is at least $n_a - \left\lfloor \frac{n_a}{n_t} \right\rfloor \cdot n_t$, and hence the master needs at most $1 + \frac{n-1}{n_a - \left\lfloor \frac{n_a}{n_t} \right\rfloor \cdot n_t} < n$ rounds and $2n - 1$ work. Thus, an algorithm analogous to Algorithm 2 has time $O(n)$ and optimal work $\Theta(n)$.

3.4 Probabilistic Worker Behavior ($\epsilon > 0$)

We are now moving to the case where the error probability of each worker is $0 < \epsilon < \frac{1}{2}$. In this case an altruistic worker may reply with an incorrect result, or a troll with a correct result with probability ϵ. Note that we do not consider the case when $\epsilon \geq \frac{1}{2}$, because in that case essentially the roles are switched. We provide algorithms that cover the full spectrum of values for the density of solutions $\rho \in (0, 1)$ and the fraction of altruistic workers $f_a \in (0, 1]$. Notice that the algorithms presented here also apply in the case where $\epsilon = 0$, but they may induce extra performance overhead.

Under this model the master receives the correct result from a randomly selected worker with probability at least $f_a(1 - \epsilon)$. We provide two different algorithms for this setting: (i) algorithm MWMIX that uses both auditing and voting, i.e. $T = \{C, V\}$, and (ii) algorithm MWVOTE that uses only voting, i.e. $T = \{V\}$, which is possible only when the density of solutions satisfy a given bound.

Note that in this section we do not present an algorithm that only relies on auditing to compute all tasks in $\mathcal{T}$ whp. This is so, because even if the set W_a of altruistic workers is known, and only these workers are used, the value returned by the execution of a task is correct only with constant $1 - \epsilon$ probability. An algorithm that does not use some form of voting will not execute a task more than once, and cannot improve this probability.

In this section we assume that the algorithms know the parameters s, f_a and ϵ. For the case that this is not true, we provide algorithm E_1 in the next section, that uses auditing to estimate them. Due to lack of space the proofs of correctness and performance of the algorithms are omitted.

Algorithm 3 The pseudo-code for algorithm MWMIX, at the master, with n workers W computing the results of n tasks in $\mathcal{T}$, where $\frac{s}{s+1} < 1 - \epsilon$ and $T = \{C, V\}$.

```
    Phase 1
 1: R[1..n] ← ∅^n                                  ▷ R[j] is the list of results from worker w_j
 2: for i = 1 : ⌈c log n⌉ do
 3:     Send challenge task t to all workers in W
 4:     Add received result from worker w_j to R[j]
 5: end for
 6: for i = 1 : n do
 7:     if # correct results in R[i] ≥ ⌈½ c log n⌉ then
 8:         U_a ← U_a ∪ {w_i}
 9:     end if
10: end for
    Phase 2
11: F[i] ← ∅                                       ▷ initially empty for all 1 ≤ i ≤ n
12: for j = 1 : ⌈k log n⌉ do
13:     for i = 1 : |U_a| : n do                   ▷ loop increments i by |U_a|
14:         Send task t_{i+k-1} to kth worker in U_a, k ∈ [1, |U_a|]
15:         Add received result for t_{i+k-1} to F[i + k − 1]
16:     end for
17: end for
18: for i = 1 : n do
19:     Results[i] ← plurality(F[i])
20: end for
21: return Results
```

3.4.1 Algorithm MWMIX: Auditing and Voting

In this section we present algorithm MWMIX, that uses a combination of auditing and voting. The general idea of the algorithm is to identify the workers in W_a, correctly *whp* and then use the estimated set U_a to compute *all* the tasks correctly *whp*. Below we describe algorithm MWMIX, with the pseudo code in Algorithm 3, using this strategy. As explained earlier, it is preferable to avoid using the *challenge* method ($T = \{C\}$) because auditing implies a computational burden on the master. Therefore, the algorithm uses auditing only to estimate U_a, and afterwards it uses voting ($T = \{V\}$) to determine the correct result for each task, i.e. $T = \{C, V\}$.

Description of algorithm MWMIX

Algorithm MWMIX consists of two phases. During the first phase, MWMIX computes an estimate of the set W_a, denoted by U_a, by using the challenge method. Phase 1 has $c \log n$ rounds, for a constant $c > 0$ that depends on ϵ (L:2). During each round the master sends out a distinct challenge task to every worker in W (L:3), and upon receiving the reponses from the workers stores the results in an array of lists $R[1], R[2], \cdots, R[n]$, where $R[i]$ denotes the list of results received from process $w_i \in W$ (L:4). Next based on the results in $R[\,]$, the ID of any worker that answered the majority of the challenge tasks correctly is included in the set U_a (L:8).

During the Phase 2, only the workers in U_a are used to compute the n tasks. Each task is

executed $\lceil k \log n \rceil$ times. The results sent back by the workers in U_a are stored in the array of lists $F[1], F[2], \cdots, F[n]$ (L:15), where the results for task t_i are stored in list $F[i]$. Finally, the master decides for every task t_i the plurality of results in $F[i]$ to be the correct result. The results of the tasks in $\mathcal{T}$ are hence stored in the array variable $Results$, where $Results[i]$ is the result of task t_i (L:19).

Correctness & Performance Analysis

Now, we prove the correctness of all the tasks *whp* and the complexity of results. In Lemma 3.1, we show that at the end of Phase 1 every altruistic worker and only altrusitic workers are included in U_a. Using this lemma, we prove Theorem 3.4 which states that every task in $\mathcal{T}$ is computed correctly *whp*.

Lemma 3.1. *In any execution of* MWMIX*, at the end of Phase 1 we have $U_a = W_a$, whp.*

Proof: Consider an execution of algorithm MWMIX. For any $w_j \in W$ we want to show that $w_j \in U_a$ iff $w_j \in W_a$. Let us examine the two directions separately.

For the *if* direction we show that if $w_j \in U_a$ then $w_j \in W_a$ *whp*, or equivalently, if $w_j \notin W_a$ then $w_j \notin U_a$ *whp*. Let us consider some worker w_j not in W_a (a troll). Let us denote by $X_1, X_2, \cdots X_k$, where $k = \lceil c \log n \rceil$, binary variables, defined such that $X_i = 1$ if the result of the i^{th} trial for w_j, stored in $R[j]$ is correct, otherwise, we have $X_i = 0$. Now observe the values of the variables $X_1, X_2, \cdots X_k$ derived from $R[j]$, after executing Lines 2-5. Note that all the results in $R[j]$ are essentially independent and identical trials by worker w_j. Hence for any i, $\mathbf{Pr}[X_i = 1] = \epsilon$. Now if $X^{w_j} = \sum_{i=1}^{k} X_i$, where the superscript w_j refers to the worker w_j under consideration, then we have $\mathbb{E}[X^{w_j}] = k\epsilon = \mu$. By Chernoff bound $\mathbf{Pr}[X^{w_j} \geq (1+\delta)\mu] \leq e^{-\frac{\mu\delta^2}{3}}$, for $\delta \in (0,1)$. Now, we choose $\delta < \frac{1}{2\epsilon} - 1$, and hence $(1+\delta)\epsilon < \frac{1}{2}$. (Observe that $\frac{1}{2\epsilon} - 1 > 0$ since $\epsilon < \frac{1}{2}$.) Then, we have $\mathbf{Pr}[X^{w_j} \geq \frac{1}{2}\lceil c \log n \rceil] \leq \mathbf{Pr}[X^{w_j} \geq (1+\delta)k\epsilon] \leq e^{-\frac{k\epsilon\delta^2}{3}} \leq \frac{1}{n^{\alpha}}$ where $\alpha > 1$ is some constant, for a c chosen sufficiently large, say c_1. The above result shows that, *wph*, the majority of the results computed by w_j are incorrect, and hence in Lines 7-9 w_j would not be added to U_a, *whp*.

For the *only if* direction, we show that if $w_j \in W_a$ then $w_j \in U_a$ *whp*. If $w_j \in W_a$ then $\mathbf{Pr}[X_i = 1] = 1 - \epsilon$. By a similar argument as above, and using Chernoff bound, $\mathbf{Pr}[X^{w_j} < \frac{1}{2}\lceil c \log n \rceil] \leq \mathbf{Pr}[X^{w_j} < (1-\delta)k(1-\epsilon)] \leq e^{-\frac{k(1-\epsilon)\delta^2}{2}} \leq \frac{1}{n^{\alpha'}}$ where $\alpha' > 1$ is a some constant, for a c sufficiently large, say c_2, and δ chosen as some value such that $\delta < 1 - \frac{1}{2(1-\epsilon)}$ (and hence $(1-\epsilon)(1-\delta) > \frac{1}{2}$).

Now, we require that both directions hold for every process in $W \backslash W_a$ and W_a respectively, whp. I.e., we want the event $\mathcal{E} \equiv \bigcap_{w \in W_a} \{X^w > \frac{1}{2}\lceil c \log n \rceil\} \bigcap \bigcap_{w \in W \backslash W_a} \{X^w < \frac{1}{2}\lceil c \log n \rceil\}$,

whp. Therefore, we have

$$\begin{aligned}\mathbf{Pr}[\mathcal{E}] &= 1 - \mathbf{Pr}[\overline{\mathcal{E}}] \\ &= 1 - \mathbf{Pr}\Big[\bigcup_{w \in W_a} \{X^w \leq \frac{1}{2}\lceil c \log n \rceil\} \bigcup \\ &\qquad \bigcup_{w \in W \setminus W_a} \{X^w \geq \frac{1}{2}\lceil c \log n \rceil\}\Big]\end{aligned}$$

Now by Boole's inequality we have

$$\begin{aligned}\mathbf{Pr}[\mathcal{E}] &\geq 1 - \sum_{w \in W_a} \mathbf{Pr}[X^w \leq \frac{1}{2}\lceil c \log n \rceil] \\ &\quad - \sum_{w \in W \setminus W_a} \mathbf{Pr}[X^w \geq \frac{1}{2}\lceil c \log n \rceil] \\ &= 1 - n\frac{1}{n^\alpha} - n\frac{1}{n^{\alpha'}} \\ &\geq 1 - \frac{1}{n^\beta}\end{aligned}$$

where $\beta > 0$. ■

Theorem 3.4. *If $\frac{s}{s+1} < 1 - \epsilon$, then Algorithm* MWMIX *computes all n tasks correctly, whp.*

Proof: By Lemma 3.1, at the end of Phase 1, $U_a = W_a$ in MWMIX *whp*, and now we show that in Phase 2 the result of all the tasks are computed correctly *whp*. In Lines 12-17 every task gets computed $\lceil k \log n \rceil$ times with workers in U_a. If $U_a = W_a$ then any of these executions returns correctly with probability $1 - \epsilon$. Let us assume that it is the case that $U_a = W_a$. Then, for any task t_j, let us define the binary random variables $X_1, X_2, \cdots X_\ell$, $\ell = \lceil k \log n \rceil$, such that, if the i^{th} execution is correct $X_i = 1$, otherwise, $X_i = 0$. Clearly, $X_1, X_2, \cdots X_\ell$ are identically and independently distributed and $\mathbb{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1 - \epsilon$. Now, let us denote $\mathbf{X}^j \triangleq X_1 + \cdots + X_\ell$ the number of correct results returned, and $\mathbf{Y}^j \triangleq \ell - \mathbf{X}^j$ the number of incorrect results returned. Since $X_1, X_2, \cdots X_\ell$ are IIDs we have by Chernoff bound, for any $\delta \in (0, 1)$, $\mathbf{Pr}[\mathbf{X}^j \leq (1 - \delta)\ell(1 - \epsilon)] \leq e^{-\frac{\delta^2(1-\epsilon)k \log n}{2}} \leq \frac{1}{n^\alpha}$ for some $\alpha > 1$, when constant k is sufficiently large.

From the assumptions of the model, the result returned by a worker from a correct computation is any of the values in $S(t)$; and similarly, the result of an incorrect computation is any of the values in $R(t)$. Therefore, if we want to guarantee that the result that is returned by algorithm MWMIX is correct, we need to guarantee that some correct value is returned more times than any incorrect value. Since there are $s = S(t)$ correct values, and ℓ returned values in total, this is guaranteed by the pigeonhole principle if $\frac{\mathbf{X}^j}{s} > \mathbf{Y}^j$, i.e., if $\mathbf{X}^j > \frac{s}{s+1}\ell$. We bound the probability

Algorithm 4 Algorithm MWVOTE, at the master process, performs n tasks using n workers for the case $\frac{s}{s+1} < f_a(1-\epsilon) + (1-f_a)\epsilon$ and $T = \{V\}$.

1: $F[i] \leftarrow \emptyset$ ▷ initially empty for all $1 \leq i \leq n$
2: **for** $i = 1$ to $\lceil k \log n \rceil$ **do** ▷ for some constant $k > 0$
3: Choose a random permutation $\pi \in \Pi_n$
4: Send each task $t_j \in \mathcal{T}$ to worker $w_{\pi(j)}$
5: Add received result from worker $w_{\pi(j)}$ to $F[j]$
6: **end for**
7: **for** $i = 1 : n$ **do**
8: $Results[i] \leftarrow plurality(F[i])$
9: **end for**
10: **return** $Results$

that this is not the case using the above Chernoff bound, defining $\delta = 1 - \frac{s}{p(s+1)}$. Hence,

$$\mathbf{Pr}[\frac{\mathbf{X}^j}{s} \leq \mathbf{Y}^j] = \mathbf{Pr}[\mathbf{X}^j \leq \frac{s}{s+1}\ell] \leq \frac{1}{n^\alpha},$$

as long as $1 - \frac{s}{p(s+1)} \in (0,1)$.

Next, by using Boole's inequality we have $\mathbf{Pr}[\bigcup_{j=1}^{n}\{\frac{\mathbf{X}^j}{s} \leq \mathbf{Y}^j\}] \leq \sum_{j=1}^{n} \mathbf{Pr}[\frac{\mathbf{X}^j}{s} \leq \mathbf{Y}^j] \leq \frac{1}{n^\gamma}$ where $\gamma = \alpha - 1 > 0$. Hence, the plurality of the results compute all the tasks correctly *whp*, conditioned to $U_a = W_a$ which also occurs whp. Hence the claim of the theorem. ■

Theorem 3.5. *Algorithm* MWMIX *runs in* $\Theta(\frac{n}{n_a}\log n)$ *synchronous rounds and performs* $\Theta(n \log n)$ *work.*

Proof: From the algorithm it is clear. ■

3.4.2 Algorithm MWVOTE: Use Voting Alone

In this section, we present algorithm MWVOTE that uses voting mechanism alone (i.e., when $T = V$) to compute all the n tasks *whp*. Algorithm MWVOTE can be used when $\frac{s}{s+1} < f_a(1-\epsilon) + (1-f_a)\epsilon$. Note here that, without identifying the altruistic workers, the probability that a randomly selected worker will reply with the correct answer for a task t is $f_a(1-\epsilon) + (1-f_a)\epsilon$; i.e. receive the correct answer from an altruistic worker with probability $1-\epsilon$ or to receive the correct answer from a troll with probability ϵ.

Description of Algorithm MWVOTE

Below we present our algorithm MWVOTE, and the pseudo-code for the algorithm is in Algorithm 4. The basic idea of MWVOTE is to exploit the fact that if a troll worker picks an answer for a task t randomly from $R(t)$ and $\frac{s}{d}$ is small, then the likelihood of an incorrect result being a majority or plurality among all the results is small. To apply this idea, the master distributes the n tasks according to a random permutation from Π_n, which is the set of all permutations

over $[n] = \{1, 2, \ldots, n\}$. In other words, if the random permutation is π then the j^{th} task t_j is delegated to worker $w_{\pi(j)}$ (L:4). The whole process is repeated $\lceil k \log n \rceil$ rounds (L:2-6). The constant k is used to tune the exponent $\ell > 0$ in the denominator of $\frac{1}{n^\ell}$, required for the high probability guarantee. The results for each task t_j is accumulated in the multiset $R[j]$. When the **for** loop in lines 2–6 terminates, the result for each task $t_j \in \mathcal{T}$ is chosen to be the one that forms a plurality of results in $R[j]$ (L:8).

Correctness & Performance Analysis

The following theorems state that algorithm MWVOTE computes all tasks correctly *whp* under the assumed case.

Theorem 3.6. *If $\frac{s}{s+1} < f_a(1-\epsilon)+(1-f_a)\epsilon$, Algorithm* MWVOTE *computes all n tasks correctly* whp.

Proof: We prove the above statement by first proving the correctness for one task *whp*, and then generalize it to all n tasks *whp* by using Boole's inequality. Consider any task $t_j \in \mathcal{T}$ computed by algorithm MWVOTE. In an execution of MWVOTE, t_j is computed $\ell \equiv \lceil k \log n \rceil$ times each in a distinct round of the algorithm.

Consider a sequence of binary random variables X_i, $1 \leq i \leq \ell$, where X_i corresponds to round i of an execution of MWMIX, where $X_i = 1$ if for task t_j in round i worker $\pi(j)$ returns a correct result, and $X_i = 0$ if it returns an incorrect result. Clearly, the random variables in $\{X_i\}$ are independent and identically distributed (IID). Now, we compute the value of $\mathbf{Pr}[X_i = 1]$. Note that according to our model, during some round i of MWVOTE, task t_j is delegated to a either worker in W_a or a worker in $W \backslash W_a$; and the workers can correctly or incorrectly compute the result. Therefore, we have $p \equiv \mathbb{E}[X_i] = \mathbf{Pr}[X_i = 1] = f_a(1-\epsilon) + (1-f_a)\epsilon$. Now, let us denote $\mathbf{X}^j \triangleq X_1 + \cdots + X_\ell$ the number of correct results returned, and $\mathbf{Y}^j \triangleq \ell - \mathbf{X}^j$ the number of incorrect results returned. Since $X_1, X_2, \cdots X_\ell$ are IIDs we have by Chernoff bound, for any $\delta \in (0,1)$ we have, $\mathbf{Pr}[\mathbf{X}^j \leq (1-\delta)\ell p] \leq e^{-\frac{\delta^2 pk \log n}{2}} \leq \frac{1}{n^\alpha}$ for some $\alpha > 1$, when constant k is sufficiently large.

From the assumptions of the model, the result returned by a worker from a correct computation is any of the values in $S(t)$; and similarly, the result of an incorrect computation is any of the values in $R(t)$. Therefore, if we want to guarantee that the result that is returned by MWVOTE is correct, we need to guarantee that some correct value is returned more times than any incorrect value. Since there are $s = S(t)$ correct values, and ℓ returned values in total, this is guaranteed by the pigeonhole principle if $\frac{\mathbf{X}^j}{s} > \mathbf{Y}^j$, i.e., if $\mathbf{X}^j > \frac{s}{s+1}\ell$. We bound the probability that this is not the case using the above Chernoff bound, defining $\delta = 1 - \frac{s}{p(s+1)}$. Hence,

$$\mathbf{Pr}[\frac{\mathbf{X}^j}{s} \leq \mathbf{Y}^j] = \mathbf{Pr}[\mathbf{X}^j \leq \frac{s}{s+1}\ell] \leq \frac{1}{n^\alpha},$$

as long as $1 - \frac{s}{p(s+1)} \in (0,1)$.

Next, by using Boole's inequality we have $\mathbf{Pr}[\bigcup_{j=1}^{n}\{\frac{\mathbf{X}^j}{s} \leq \mathbf{Y}^j\}] \leq \sum_{j=1}^{n} \mathbf{Pr}[\frac{\mathbf{X}^j}{s} \leq \mathbf{Y}^j] \leq \frac{1}{n^\gamma}$ where $\gamma = \alpha - 1 > 0$. Hence, the plurality of the results compute all the tasks correctly *whp*. ■

Theorem 3.7. *Algorithm* MWVOTE *runs in* $\Theta(\log n)$ *synchronous rounds and performs* $\Theta(n \log n)$ *work.*

Proof: Clear from the algorithm. ■

Remark 3.1. *Note that the algorithms presented in this section have a log factor overhead on the optimal work (Theorem 3.1). According to [78] a factor of log work overhead is the least amount of work required per task when voting is used. Thus, we can safely conclude on the optimality of both algorithms* MWMIX *and* MWVOTE.

3.5 Algorithm E_1: Tightly Estimating f_a and ϵ

As shown in Section 3.4, algorithms MWMIX and MWVOTE are possible only if we know parameters of the system, like the probability of error ϵ, the fraction of altruistic workers f_a, and the number of correct results s, to check applicability. In this section, we assume s is known, but that neither the value of f_a and ϵ are known *a priori*. (Note that it is reasonable to assume that the number of correct answers s is known.) Hence, we provide an algorithm to estimate f_a and ϵ, *whp*. As a byproduct, the algorithm also estimates $f_a(1-\epsilon) + (1-f_a)\epsilon$, *whp*. Our goal is to estimate all these values, with user defined bounds, in a manner called (ε, δ)-approximation. By choosing $\varepsilon, \delta \in O(\frac{1}{n^c})$ for some $c > 0$ we can provide a tight estimate of the value within a $\pm\varepsilon$ factor and whp (greater than $1 - \delta$).

Formally, let Z be a random variable distributed in the interval $[0, 1]$ with mean μ_Z. Let $Z_1, Z_2, \ldots$ be independently and identically distributed according to the Z variable. We say that an estimate $\tilde{\mu}_Z$ is an (ε, δ)-approximation of μ_Z if $\mathbf{Pr}[\mu_Z(1-\varepsilon) \leq \tilde{\mu}_Z \leq \mu_Z(1+\varepsilon)] > 1 - \delta$. Estimating the value of μ_Z may be done by collecting *sufficient* samples and selecting the majority as the outcome. However, such a solution is not feasible if the number of samples are not known *a priori*.

The Stopping Rule Algorithm

Algorithm 6 is an algorithm for calculating an (ε, δ)-approximation of the desired parameters, where the error tolerance bounds δ and ε are $O(\frac{1}{n^c})$, for some $c > 0$. The core idea behind E_1 is based on the Stopping Rule Algorithm (*SRA*) of Dagum *et al.* [43]. For completeness we reproduce in Algorithm 5 the *SRA* for estimating the mean of a random variable with support in $[0, 1]$, with (ε, δ)-approximation. *SRA* computes an (ε, δ)-approximation with an optimal number of samplings, within a constant factor [43]. Thus *SRA*-based method provides substantial computational savings. Let us define $\lambda = (e - 2) \approx 0.72$ and $\Gamma = 4\lambda \log(\frac{2}{\delta})/\varepsilon^2$. Now, Theorem 3.8

Algorithm 5 The Stopping Rule Algorithm (*SRA*) for estimating μ_Z.

input parameters: (ε, δ) with $0 < \varepsilon < 1, \delta > 0$
1: Let $\Gamma = 4\lambda \log(\frac{2}{\delta})/\varepsilon^2$ ▷ $\lambda = (e-2) \approx 0.72$
2: Let $\Gamma_1 = 1 + (1+\varepsilon)\Gamma$
3: **initialize** $N \leftarrow 0, S \leftarrow 0$
4: **while** $S < \Gamma_1$ **do**
5: $\quad N \leftarrow N + 1$
6: $\quad S \leftarrow S + Z_N$
7: **end while**
8: **return** $\tilde{\mu}_Z \leftarrow \frac{\Gamma_1}{N}$

(slightly modified, from [43]) tells us that *SRA* provides us with an (ε, δ)-approximation with the number of trials within $\frac{\Gamma_1}{\mu_Z}$ *whp*, where $\Gamma_1 = 1 + (1+\varepsilon)\Gamma$.

Theorem 3.8. *[Stopping Rule Theorem] Let Z be a random variable in $[0,1]$ with $\mu_Z = \mathbb{E}[Z] > 0$. Let $\tilde{\mu}_Z$ be the estimate produced and let N_Z be the number of experiments that SRA runs with respect to Z on inputs ε and δ. Then, (i) $\mathbf{Pr}[\mu_Z(1-\varepsilon) \leq \tilde{\mu}_Z \leq \mu_Z(1+\varepsilon)] > 1-\delta$; (ii) $\mathbb{E}[N_Z] \leq \frac{\Gamma_1}{\mu_Z}$, and (iii) $\mathbf{Pr}[N_Z > (1+\varepsilon)\frac{\Gamma_1}{\mu_Z}] \leq \frac{\delta}{2}$.*

Description of Algorithm E_1

The idea behind algorithm E_1 is to sample two binary random variables: (i) $Z^1 \in \{0,1\}$, whose mean is "close" to f_a; and (ii) $Z^2 \in \{0,1\}$, whose mean is $f_a(1-\epsilon) + (1-f_a)\epsilon$. Then E_1 creates (ε, δ)-approximation estimates for both of these means, using the *SRA* algorithm for $\delta, \varepsilon \in O(\frac{1}{n^c})$, for some $c > 0$. Using these estimates, it solves for a (ε, δ)-approximation for the different parameters. Below we explain the sampling process.

Z^1 is defined as follows: the master randomly picks a worker w from W, sends ℓ (a positive integer, explained later) challenges to w, and collects and verifies the results. If the majority of the results R are correct then $Z^1 = 1$, otherwise $Z^1 = 0$. We use $CorrMaj(R)$ to denote that the majority of the results in R are correct. Clearly,

$$\begin{aligned}
\mathbb{E}[Z^1] &= \mathbf{Pr}[w \in W_a] \cdot \mathbf{Pr}[CorrMaj(R) | w \in W_a] \\
&\quad + \mathbf{Pr}[w \notin W_a] \cdot \mathbf{Pr}[CorrMaj(R) | w \notin W_a] \\
&= f_a \cdot \mathbf{Pr}[CorrMaj(R) | w \in W_a] \\
&\quad + (1 - f_a) \cdot \mathbf{Pr}[CorrMaj(R) | w \notin W_a]
\end{aligned}$$

Next, by exploiting the fact that $\epsilon < \frac{1}{2} - \zeta$, where $\zeta > 0$ is a constant, we choose ℓ appropriately, such that, $\mathbf{Pr}[CorrMaj(R) | w \in W_a] \approx 1$ (i.e., $1 - O(\frac{1}{n^c})$, for some $c > 0$) and $\mathbf{Pr}[CorrMaj(R) | w \notin W_a]$ becomes very small (i.e., $O(\frac{1}{n^c})$, for some $c > 0$). Hence $\mathbb{E}[Z^1]$ approximated suitably enough $f_a = \mathbf{Pr}[w \in W_a]$. Lines 4–15 for algorithm E_1 implements the *SRA* algorithm to estimate $\mathbb{E}[Z^1]$.

Z^2 is defined as follows: the master randomly picks a worker w from W, assigns a challenge to w, and verifies the reported result. If the result is correct then $Z^2 = 1$, otherwise $Z^2 = 0$. Note

that

$$\begin{aligned}\mathbb{E}[Z^2] &= \mathbf{Pr}[w \in W_a] \cdot \mathbf{Pr}[result\ is\ correct | w \in W_a] \\ &\quad + \mathbf{Pr}[w \notin W_a] \cdot \mathbf{Pr}[result\ is\ correct | w \notin W_a] \\ &= f_a(1-\epsilon) + (1-f_a)\epsilon.\end{aligned}$$

Lines 16–24 for algorithm E_1 implement the *SRA* algorithm to estimate $\mathbb{E}[Z^2]$.

Algorithm 6 Algorithm E_1 to estimate f_a, ϵ, and $f_a(1-\epsilon) + (1-f_a)\epsilon$.

1: Let $\delta = \frac{1}{n^c}$ and $\varepsilon = \frac{1}{n^c}$ for $c > 0$
2: Let $\Gamma = (4\lambda \log(\frac{2}{\delta}))/\varepsilon^2$ and $\Gamma_1 = 1 + (1+\varepsilon)\Gamma$
3: Let $\ell = \lceil k \log n \rceil$, for some $k > 0$
4: $N \leftarrow 0, S \leftarrow 0$
5: **while** $S < \Gamma_1$ **do**
6: $\quad N \leftarrow N + 1$
7: $\quad$ pick a worker w randomly uniformly from W
8: $\quad$ **for** $i = 1$ to ℓ **do**
9: $\quad\quad$ send challenge task t_i to w
10: $\quad\quad$ $R[i] \leftarrow$ result received from w
11: $\quad$ **end for**
12: $\quad$ **if** $CorrMaj(R)$ **then** $Z_N^1 \leftarrow 1$ **else** $Z_N^1 \leftarrow 0$ **end if**
13: $\quad S \leftarrow S + Z_N^1$
14: **end while**
15: $\tilde{p} \leftarrow \frac{\Gamma_1}{N}$
16: $N \leftarrow 0, S \leftarrow 0$
17: **while** $S < \Gamma_1$ **do**
18: $\quad N \leftarrow N + 1$
19: $\quad$ pick a worker w randomly uniformly from W
20: $\quad$ send challenge task to w
21: $\quad$ **if** result received from w is correct **then** $Z_N^2 \leftarrow 1$ **else** $Z_N^2 \leftarrow 0$ **end if**
22: $\quad S \leftarrow S + Z_N^2$
23: **end while**
24: $\tilde{q} \leftarrow \frac{\Gamma_1}{N}$
25: **return** $\left(\tilde{p}, \frac{\tilde{q}-\tilde{p}}{1-2\tilde{p}}, \tilde{q}\right)$

Analysis of the Algorithm

In the following theorem we state that E_1 provides suitable approximation for the different parameters of the system.

Theorem 3.9. *The estimates $\tilde{p}$, $\frac{\tilde{q}-\tilde{p}}{1-2\tilde{p}}$, and $\tilde{q}$ returned by E_1 are (ε, δ)-approximations of the parameters f_a, ϵ, and $f_a(1-\epsilon) + (1-f_a)\epsilon$, respectively, where $\varepsilon, \delta \in O(\frac{1}{n^\gamma})$, for some $\gamma > 0$.*

Proof: We first compute an estimate for the quantity

$$\begin{aligned}p &\triangleq f_a \cdot \mathbf{Pr}[CorrMaj(R) | w \in W_a] \\ &\quad + (1-f_a) \cdot \mathbf{Pr}[CorrMaj(R) | w \notin W_a]\end{aligned}$$

Now we compute $\mathbf{Pr}[CorrMaj(R)|w \in W_a]$ and $\mathbf{Pr}[CorrMaj(R)|w \notin W_a]$.

Case $\mathbf{Pr}[CorrMaj(R)|w \in W_a]$: Suppose $w \in W_a$ and denote $\ell \triangleq \lceil k \log n \rceil$, where k is some constant whose value is decided later. Let us define binary random variables $X_1, X_2, \cdots, X_\ell$, such that, $X_i = 1$ if the i^{th} challenge task computed by w is found to be *correct* by the master, otherwise, $X_i = 0$; we also define the random variable $\mathbf{X} \triangleq X_1 + \cdots + X_\ell$. Now, since $\epsilon < \frac{1}{2} - \zeta$ then for any $0 < \delta < 1 - \frac{1}{1+2\zeta}$, we have $(1-\delta)(1-\zeta) > \frac{1}{2}$, and using Chernoff bound as the X_is are independent and identically distributed, then we have

$$\begin{aligned}
\mathbf{Pr}[\mathbf{X} \leq \frac{\lceil k \log n \rceil}{2}] &\leq \mathbf{Pr}[\mathbf{X} \leq (1-\delta)\lceil k \log n \rceil (1-\epsilon)] \\
&\leq e^{-\frac{\ell(1-\epsilon)\delta^2}{2}} \\
&\leq e^{-\frac{\delta^2 \lceil k \log n \rceil}{4}} \\
&\leq \frac{1}{n^a},
\end{aligned}$$

where k is chosen appropriately and the constant $a > 0$ depends on k.

Case $\mathbf{Pr}[CorrMaj(R)|w \notin W_a]$: Now consider $w \notin W_a$, and we define a set of ℓ binary variables $Y_1, \cdots, Y_\ell$ such that, $Y_i = 1$ if the i^{th} task computed by w is found to be *incorrect*, otherwise $Y_i = 0$; and we define the random variable $\mathbf{Y} \triangleq Y_1 + \cdots + Y_\ell$. Now we evaluate the following bound, by suitable picking some $0 < \delta < 1 - \frac{1}{1+2\zeta}$ and a constant k, then for sufficiently large n we have a constant $b > 0$, such that $\mathbf{Pr}[\mathbf{Y} \geq \frac{1}{2}\lceil k \log n \rceil] = 1 - \mathbf{Pr}[\mathbf{Y} \leq \frac{1}{2}\lceil k \log n \rceil] \geq 1 - \mathbf{Pr}[\mathbf{Y} \leq (1-\delta)\lceil k \log n \rceil (1-\epsilon)] \geq 1 - e^{-\frac{\ell(1-\epsilon)\delta^2}{2}} \geq 1 - e^{-\frac{\delta^2 \lceil k \log n \rceil}{4}} \geq 1 - \frac{1}{n^b}$.

Now using the above relations we have the following two bounds $\mathbf{Pr}[CorrMaj(R)|w \in W_a] = \mathbf{Pr}[\mathbf{X} \leq \frac{1}{2}\lceil k \log n \rceil] = 1 - \mathbf{Pr}[\mathbf{X} > \frac{1}{2}\lceil k \log n \rceil] \geq 1 - \frac{1}{n^a}$, and $\mathbf{Pr}[CorrMaj(R)|w \notin W_a] = 1 - \mathbf{Pr}[not\ CorrMaj(R)|w \notin W_a] = 1 - \mathbf{Pr}[Y \geq \frac{1}{2}\lceil k \log n \rceil] \leq \frac{1}{n^b}$ Using the above two bounds we have $p \geq f_a\left(1 - \frac{1}{n^a}\right) + (1 - f_a)0 \geq f_a\left(1 - \frac{1}{n^a}\right)$ and $p \leq f_a + (1 - f_a)\frac{1}{n^b} \leq f_a\left(1 + \frac{1}{n^b}\right)$, and without loss of generality, we can adjust the constant k such that we can choose a and b as equal $f_a\left(1 - \frac{1}{n^a}\right) \leq p \leq f_a\left(1 + \frac{1}{n^a}\right)$.

Now, observe that the variable Z_N^1 (line 12) in algorithm E_1, is essentially the counter part of Z_N in Fig. 5. In each iteration Z_N^1 is sampled independently of and identically as other iterations, which holds also for Z_N in *SRA*. Also, $Z_N^1 \in [0,1]$, of course all of the probability is concentrated in $\{0,1\} \subseteq [0,1]$ and $\mathbb{E}[Z_N^1] = p$. So, we observe that in Lines 4–15 in Fig. 5 essentially implements the *SRA* on the random variable Z_N^1 . Therefore, by Theorem 3.8, we have $\mathbf{Pr}[\left(1 - \frac{1}{n^\alpha}\right)p \leq \tilde{p} \leq \left(1 + \frac{1}{n^\alpha}\right)p] \geq 1 - \frac{1}{n^\beta}$, next using the above bound for p we get $\mathbf{Pr}[\left(1 - \frac{1}{n^\alpha}\right)\left(1 - \frac{1}{n^a}\right)f_a \leq \tilde{p} \leq \left(1 + \frac{1}{n^\alpha}\right)\left(1 + \frac{1}{n^a}\right)f_a] \geq 1 - \frac{1}{n^\beta}$, which can be simplified, for some $\gamma > 0$ as $\mathbf{Pr}[\left(1 - \frac{1}{n^\gamma}\right)f_a \leq \tilde{p} \leq \left(1 + \frac{1}{n^\gamma}\right)f_a] \geq 1 - \frac{1}{n^\beta}$.

Now suppose we define $q \triangleq f_a(1-\epsilon) + (1-f_a)\epsilon$, then clearly we can solve for ϵ as $\epsilon = \frac{q - f_a}{1 - 2f_a}$. Now, observe that the variable Z_N^2 (line 21) in algorithm E_1, is essentially the counter part of Z_N in Fig. 5. As in the case for Z_N^1 each iteration Z_N^2 is sampled independently of and identically as other iterations, which holds also for Z_N in *SRA*. Therefore, arguing as in Z_N^1, so we observe

that in lines 16–24 in Fig. 5 essentially implements the *SRA* on the random variable Z_N^2 . Also, $Z_N^2 \in [0,1]$ and all probability is concentrated in $\{0,1\} \subseteq [0,1]$ and $\mathbb{E}[Z_N^2] = q$. Therefore, by Theorem 3.8, we have the following bound for the variable $\tilde{q}$, $\mathbf{Pr}[\left(1 - \frac{1}{n^\alpha}\right) q \leq \tilde{q} \leq \left(1 + \frac{1}{n^\alpha}\right) q]$ $\geq 1 - \frac{1}{n^\beta}$.

Now we show that the quantity $\frac{\tilde{q}-\tilde{p}}{1-2\tilde{p}}$ is a very close approximation for ϵ. Observe that by using the appropriate bounds for $\tilde{q}$ and $\tilde{p}$, derived above, we have $\frac{\tilde{q}-\tilde{p}}{1-2\tilde{p}} \leq \frac{q\left(1+\frac{1}{n^\alpha}\right)-f_a\left(1-\frac{1}{n^\alpha}\right)}{1-2f_a\left(1+\frac{1}{n^\alpha}\right)} =$ $\frac{(q-f_a)+(q+f_a)\frac{1}{n^\alpha}}{(1-2f_a)-\frac{2f_a}{n^\alpha}} \leq \left(\frac{q-f_a}{1-2f_a}\right)\left(1+\frac{1}{n^\gamma}\right) = \epsilon\left(1+\frac{1}{n^\gamma}\right)$. Similarly, we can show that $\epsilon\left(1-\frac{1}{n^\gamma}\right) \leq$ $\frac{\tilde{q}-\tilde{p}}{1-2\tilde{p}}$ Now, combining the above we have $\mathbf{Pr}[\epsilon\left(1-\frac{1}{n^\gamma}\right) \leq \frac{\tilde{q}-\tilde{p}}{1-2\tilde{p}} \leq \epsilon\left(1+\frac{1}{n^\gamma}\right)] \geq 1 - \frac{1}{n^\gamma}$. ■

Theorem 3.10. *The number of rounds or the work for algorithm E_1 is $n^c \log n$ for $c > 0$* whp.

Proof: The number of times the **while** loop iterates is the value of N at the end of the looping. The value of N at the end of any of the **while** loop is $n^c \log n$ *whp*, which can be proved by substituting $\frac{1}{n^c}$ for δ and ε in the Γ_1 (see Algorithm 6) and applying Theorem 3.8(iii). Now, in the first **while** the **for** loop runs for $\Theta(\log n)$ iterations. Therefore, the rounds and work are $O(n^c \log^2 n)$, *whp*. ■

Chapter 4

A Reinforcement Learning Approach

4.1 Introduction

In this chapter we consider once more the problem of unreliable task computations over the Internet following the master-worker model. Recall that, in the previous chapter we modelled the worker's unreliable behavior through the error probability model, considering the presence of altruistic and troll workers, that can deviate from their true nature by an error probability ϵ. We assumed that ϵ and the fraction of altruistic workers is known, or can be estimated with high probability (whp), and we evaluated two reliability techniques, auditing and voting, under the assumption that a density of solutions exists. Reliability techniques are evaluated in terms of time and work as described in the previous chapter.

Having a good intuition on the pros and cons of each reliability technique we take a step forward towards a mechanism that assumes no knowledge about the workers distribution in the system. The goal of this mechanism will be to eventually guarantee that the master will always be receiving the correct task result with probability one and with minimal cost. In the previous chapter we modelled the diversity of the workers' behavior though the error probability. In this chapter we attempt a different modelling approach. We still assume that the workers might have malicious behavior by reporting an incorrect value. In the previous chapter we were referring to these workers as trolls. Here we choose to call them malicious since we assume that they have an intelligent strategy that always provides them with the incorrect value and more over we assume a worse case form of collusion where all malicious workers reply with the same incorrect value. Additionally, like in the previous chapter we assume the presence of altruistic workers, assuming that they will always be reporting the correct task value. One of the main contributions of this chapter is that we model the diversity in the workers behavior though "rationality". That is we assume that an unknown ratio of workers will behave in a rational manner. A rational worker, behaves in such a way choosing the appropriate strategy that will maximize its benefit. Thus, if a master where to interact with the same worker multiple times, then potentially the behavior of the rational worker could be "reinforced" to serve the master's goal.

Given the above, we consider Internet-based task computations with a unique correct solution, where a master process sends tasks, across the Internet, to a fix number of 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 some 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 task results. In particular, we develop and analyze an algorithmic mechanism based on reinforcement learning to provide rational workers with the necessary incentives to eventually become truthful. Our designed mechanism uses, an auditing technique where the master computes the correct task result by itself to reward and punish the workers accordingly. The master deals with the malicous workers behavior through a reputation mechanism. Moreover, to address the issue of the workers unavailability we design a reputation scheme together with a mechanism that chooses the workers to participate in a task computation from the whole set of workers (we call it here "pool" of workers).

Chapter Organization

The rest of the Chapter is organized in four Sections:

- Section 4.2 presents the general model that all the mechanisms designed in the chapter follow. In this section we present the general master-worker model, the tasks considered as well as the worker types considered. Moreover, we present the master's auditing technique and the incentives used. Finally, we formally present the master's goal, which we call eventual correctness.

- Section 4.3 presents a first approach mechanism that considers only the existence of rational workers. The mechanism is complemented by an analytical part that identifies the conditions under which truthful behavior can be ensured, and bounds the expected convergence time to that behavior. The illustrative simulations presented show trade-offs among a number of system parameters.

- The mechanism presented in Section 4.3 is complemented and enhanced by a reputation-based scheme presented in Section 4.4 that coops with the existence of malicious workers. In this section, we upgrade the model by considering the presence of malicious and altruistic workers beside the presence of rational workers. The system analysis gives provable guarantees under which truthful behavior can be ensured. We observe the behavior of the mechanism through simulations that reveal interesting trade-offs between various metrics and parameters. The correlation among cost and convergence time to a truthful behavior is shown and the four reputation schemes designed are assessed against the tolerance to cheaters.

- Finally, in Section 4.5 we vary our current mechanism approach (assuming the master assigns tasks to a fixed predefined set of workers) by allowing the master to select the most reputable and responsive workers from a pool of workers. The general model is being redefined in some aspects and a responsiveness reputation scheme is presented to coop with the unavailability of the workers. Our analysis proves sufficient conditions for eventual correctness under different reputation schemes, and is complemented by simulations that show interesting trade-offs among the different reputation types, workers availability and master's cost.

4.2 General Model

The mechanisms described in the rest of the chapter consider a generic model that captures Internet-based task computations and the behavior of the participating components, that is the master and the workers. Below we describe in detail, the master-worker model, the nature of the computational task considered as well as the worker's behavior, together with the auditing and incentive techniques used. Finally, we introduce the concept of eventual correctness, that describes the goal pursued by the master.

Master-Worker Model

We consider a system consisting of a set G of voluntary workers, that is workers declaring that they are willing to perform tasks computations. The set G is broken down into disjoint sets W_j of size n forming the group of workers receiving a replica of the same task. For simplicity we will focus at only one such set of workers named W. Hence we consider a master and a set W of n workers (with out loss of generality, we assume that n is odd). The computation is broken into *rounds*, and in each round the master sends a task to the workers to compute and return the result. Based on the workers' replies, the master must decide which is the value most likely to be the correct result for this round.

Tasks

The tasks considered in this chapter have a unique correct solution. Although such an assumption might seem limiting, there are plenty of computations where the correct solution is unique: e.g., any mathematical function. In particular from the crowdsoucing perspective these computations can be questions with a single correct solution. While in BOINC-operated applications we are considering the output [130] of an executable file.

Worker Types

Based on the existing literature discussed in Chapter 2 we consider that workers can be categorized in three types: *rational, altruistic* and *malicious*, following what we call a *rationality model*. This categorizations corresponds to distinct behaviors from the workers that are described below:

Rational: Following Abraham et al. [4], and Shneidman and Parkes [124], 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 to cheat depending on which strategy maximizes its utility.

Altruistic: They have a predefined behavior, to always be honest, by returning a correct value.

Malicious: They have a predefined behavior, to always cheat, by returning an incorrect value.

Notice that while altruistic and malicious workers have a predefined behavior, to always be honest or 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. This probability is not fixed and the worker adjusts it over the course of the computation. The master is not aware of the worker types, neither of a distribution of types. Thus, the mechanisms presented in this section do not rely on any statistical information.

While workers make their decision individually and with no coordination, following [118] and [51], we assume that all the workers that cheat in a round return the same incorrect value; 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. This can be seen as a weak form of collusion.

For simplicity, unless otherwise stated, we assume that workers do not change their type over time. In practice it is possible that changes occur. For example, a rational worker might become malicious due to a bug, or a malicious worker (e.g., a worker under the influence of a virus) become altruistic (e.g., if an antivirus software reinstates it). If this may happen, then all our results still apply for long enough periods between two changes. In the evaluation of the designed mechanisms through simulations we consider scenarios where the workers change their type dynamically.

Auditing and Incentives

In the presence of rational workers the master needs to induce their correct behavior, that is to be honest. Thus, when necessary (as instructed by it's algorithm) the master employs *auditing*

and *reward/punishment* schemes to influence the behavior of the rational worker.

The master, in a round, might decide to audit the responses of the workers, at a cost. By auditing we mean 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, 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 majority or weighted majority (see Section 4.3 and Sections 4.4 & 4.5 respectively) of the replies received and punishes no one. Unlike in the previous chapter, here we assume that the master has the potential of computing the task itself instead of using challenges. This is a valid assumption since we assume solutions with a unique correct solution, and as a result the master can accurately compute the solution of a task.

$WP_{\mathcal{C}}$	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

Table 4.1: **Payoffs. The parameters are non-negative.**

We assume that each rational worker i has an **aspiration** a_i (the same in all rounds) which is the minimum benefit it expects to obtain in a round. Given this assumption, the master considers three **payoff** parameters described in in Table 4.1 to give incentives to the rational workers to reply with a correct value. 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. 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 tries to achieve the goal of eventual correctness (see below). 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. 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*.

4.3 Presence of Rational Workers

4.3.1 Introduction

In this section we assume that all workers taking part in the computation are following a rational behavior. This means that the workers will report an incorrect results if this will increase their benefit. Making the initial assumption that the workers are all rational will allow us to study and design mechanisms that exploit the multiple interactions of the master with the same workers offering incentives that will drive, over time, the workers to an honest behavior.

To design such a mechanism that benefits from the repeated interaction among master and workers, we introduce the concept of *evolutionary dynamics* (widely used under the biological and social perspective) and apply it to Internet-based master-worker task computing. More specifically, we employ *reinforcement learning* [28, 128] 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 likelihood of reusing the strategy just chosen, and negative payoffs reduce it. Payoffs are seen as parameterizations of players' responses to their experiences. Empirical evidence [21, 29] suggests that reinforcement learning is more plausible with players that have information only on the payoffs they receive; i.e., they do not have knowledge of the strategies involved. This model of learning fits nicely Internet-based computing systems since each worker has no information about the master and the other workers' strategies and it does not know the set of strategies that led to the payoff it received. The workers have information only about their strategies and the payoffs that they receive. 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.

Section Overview

- We develop and analyze a mechanism in Subsection 4.3.2 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 their responses with a certain probability $p_{\mathcal{A}}$. Depending on whether it audits or not, it applies a different reward/punishment scheme, and adjusts the probability $p_{\mathcal{A}}$ for the next round (also known as the next task execution). Similarly, in a round, each worker i decides, with a certain probability p_{Ci}, whether it will report an incorrect result or it will truthfully compute and report the correct task result. Depending on the outcome of its decision, measured by the increase or the decrease of the worker's *utility*, the worker adjusts its probability p_{Ci} for the next round.

- In Subsection 4.3.3 we show necessary and sufficient conditions under which the mechanism ensures *eventual correctness*. That is, we establish 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 (with respect to their utility). Eventual correctness can be viewed as a form of *Evolutionary Stable Strategy* [45, 57] as studied in Evolutionary Game Theory (EGT) [132]: even if a "mutant" worker decides to change its strategy to cheating, it will soon be brought back to an honest strategy.

- In Subsection 4.3.3, 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, in Subsection 4.3.4 for a variety of parameter combinations likely to occur in practice.

4.3.2 Algorithmic Mechanism

We now present the algorithms that the master and the workers follow.

Master's Algorithm (Alg. 7)

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, after 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 accordingly.

Algorithm 7 Master's Algorithm

```
1  p_A ← x, where x ∈ [p_A^min, 1]
2  for r ← 1 to ∞ do
3    send a task T to all workers in W
4    upon receiving all answers do
5      audit the answers with probability p_A
6      if the answers were not audited then
7        accept the majority
8      else
9        p'_A ← p_A + α_m(cheaters(r)/n − τ)
10       p_A ← min{1, max{p_A^min, p'_A}}
11     ∀i ∈ W : pay/charge Π_i to worker i
```

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 pos-

sesses. 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 determine the number of cheaters; we denote by $cheaters(r)$ the number of cheaters in round r. Then, the master adapts the auditing probability $p_{\mathcal{A}}$ according to this number. 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 change in the number of cheaters, 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 (Alg. 8).

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.

Algorithm 8 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
  $p'_{Ci} \leftarrow p_{Ci} - \alpha_w(\Pi_i - a_i)S_i$
  $p_{Ci} \leftarrow \max\{0, \min\{1, p'_{Ci}\}\}$

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 (see the discussion in [128]; the learning rate is called step-size there); note that our analysis can be adjusted to accommodate also workers with different learning rates. We choose the value of α_w so that $\alpha_w(a_i + WP_{\mathcal{C}}) < 1$, $\forall i \in W$. Otherwise, the system could enter in an oscillating condition where some nodes alternate p_C between 0 and 1 never

converging to a stable state, which is necessary to guarantee reliability.

4.3.3 Analysis

In this subsection we analyze the mechanism presented in Section 4.3.2. We model the evolution of the mechanism as a Markov chain, and we prove necessary and sufficient conditions for achieving eventual correctness. We provide analytical evidence that convergence to eventual correctness can be reached rather quickly. Observe in Algorithms 7 and 8 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.

The Mechanism as a Markov Chain

We analyze the evolution of the master-workers system as a Markov chain. To do so, we first define 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 $(p^r_{\mathcal{A}}, p^r_{C1}, p^r_{C2}, \ldots, p^r_{Cn})$. Observe from Algorithms 7 and 8 that any state $(p_{\mathcal{A}}, p_{C1}, p_{C2}, \ldots, p_{Cn})$ in which $p_{\mathcal{A}} \in [p^{min}_{\mathcal{A}}, 1]$ 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 $p^r_{\mathcal{A}} \in [p^{min}_{\mathcal{A}}, 1]$ and $p^r_{Ci} \in [0, 1]$, we can describe the transition function of the Markov chain in detail. For each subset of workers $F \subseteq W$, $P(F) = \prod_{j \in F} p^{r-1}_{Cj} \prod_{k \notin F} (1 - p^{r-1}_{Ck})$ is the probability that the set of cheaters is exactly F in round r. Then, we have the following.

- With probability $p^{r-1}_{\mathcal{A}} \cdot P(F)$, the master audits when the set of cheaters is F, and then,

  (0) the master updates $p_{\mathcal{A}}$ as $p^r_{\mathcal{A}} = p^{r-1}_{\mathcal{A}} + \alpha_m(|F|/n - \tau)$, and

  (1) each worker $i \in F$ updates p_{Ci} as $p^r_{Ci} = p^{r-1}_{Ci} - \alpha_w(a_i + WP_{\mathcal{C}})$,

  (2) each worker $i \notin F$ updates p_{Ci} as $p^r_{Ci} = p^{r-1}_{Ci} + \alpha_w(a_i - (WB_{\mathcal{Y}} - WC_{\mathcal{T}}))$.

- With probability $(1 - p^{r-1}_{\mathcal{A}})P(F)$, 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 $p^r_{Ci} = p^{r-1}_{Ci} + \alpha_w(WB_{\mathcal{Y}} - a_i)$,

  (4) if $|F| < n/2$ then $p^r_{Ci} = p^{r-1}_{Ci} - \alpha_w \cdot a_i$,

  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.

Conditions for Eventual Correctness

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. Lemma 4.1 shows that, if $p_A = 0$ then 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 4.2 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 4.3 shows that the honest set including all the states in which all covered workers are honest is closed, which proves (i). Lemma 4.4 shows that any honest set where some covered worker is not honest is not closed, and Lemma 4.5 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 4.1.

Lemma 4.1. *Consider any set of workers $Z \subseteq W$ such that $\forall i \in Z : WB_{\mathcal{Y}} \geq a_i$. If $|Z| > n/2$, then the set of states $S = \{(p_{\mathcal{A}}, p_{C1}, \ldots, p_{Cn}) | (p_{\mathcal{A}} = 0) \wedge (\forall w \in Z : p_{Cw} = 1)\}$, is a closed cheat set.*

Proof: Observe first that each state in S is a cheat state, since the master does not audit and a majority of workers cheat. From transition (3) it can be seen that, if the chain is in a state of the set S before round r, for each worker $i \in Z$, $p^r_{Ci} \geq p^{r-1}_{Ci} = 1$ holds. In addition, $p_{\mathcal{A}}$ does not change. Hence, once the chain has reached a state in the set S, it will move only to states in the set S. ■

As already mentioned, from this lemma, it is concluded that $p_{\mathcal{A}} > 0$ is required for eventual correctness. From now on, it is assumed that in all rounds $p_{\mathcal{A}} \geq p^{min}_{\mathcal{A}} > 0$.

Lemma 4.2. *If there exists a set of workers $Z \subseteq W$ such that $|Z| > n/2$ and $\forall i \in Z : WB_{\mathcal{Y}} < a_i + WC_{\mathcal{T}}$ then no honest set is closed.*

Proof: Recall that we choose the value of α_w so that $\forall i \in W : \alpha_w(a_i + WP_{\mathcal{C}}) < 1$. Consider any starting state, which by assumption is an honest state, $S = \{(p_{\mathcal{A}}, p_{C1}, \ldots, p_{Cn}) | \exists Y \subseteq W : (|Y| > n/2) \wedge (\forall w \in Y : p_{Cw} = 0)\}$.

Let the set Z be divided in three sets depending on whether the workers are honest ($p_C = 0$), cheaters ($p_C = 1$) or cheat with a probability between zero and one ($0 < p_C < 1$). We denote these sets by Z_0, Z_1 and Z_b respectively. In the next round the master audits (possible since $p_{\mathcal{A}} > 0$), workers in Z_0 and Z_b do not cheat and workers in Z_1 cheat. Then from transition (2) all workers in Z_0 and Z_b increase their probability of cheating. From transition (1) all workers in Z_1 decrease their cheating probability by $\alpha_w(a_i + WP_{\mathcal{C}})$. Since all workers in Z_1 are cheater workers ($p_C = 1$) and $\alpha_w(a_i + WP_{\mathcal{C}}) < 1$, after this round their cheating probability is larger than 0. Hence, for all workers in Z their cheating probability is larger than 0 and the new state is not honest. ■

Lemma 4.3. *Consider any set of workers $Z \subseteq W$ such that $\forall i \in Z : WB_{\mathcal{Y}} \geq a_i + WC_{\mathcal{T}}$ and $\forall j \notin Z : WB_{\mathcal{Y}} < a_j + WC_{\mathcal{T}}$. If $|Z| > n/2$, then the set of states $S = \{(p_{\mathcal{A}}, p_{C1}, \ldots, p_{Cn}) | \forall w \in Z : p_{Cw} = 0\}$, is an honest closed set.*

Proof: Consider any round r before which the state of the chain is $s \in S$. Given that $|Z| > n/2$, at round r we have $cheaters(r) < n/2$. Then, for all workers in Z, the transition function is either (2) or (6), depending on whether the master audits or not. Then, given that $WB_{\mathcal{Y}} \geq a_i + WC_{\mathcal{T}}$ for all workers in Z, their probability of cheating after round r is still 0. Hence, the claim follows. ■

Lemma 4.4. *Consider any set of workers $Z \subseteq W$ such that $\forall i \in Z : WB_{\mathcal{Y}} \geq a_i + WC_{\mathcal{T}}$ and $\forall j \notin Z : WB_{\mathcal{Y}} < a_j + WC_{\mathcal{T}}$. Then, for any set of states $S = \{(p_{\mathcal{A}}, p_{C1}, \ldots, p_{Cn}) | \exists Y \subseteq W : (|Y| > n/2) \wedge (\forall w \in Y : p_{Cw} = 0) \wedge (Z \nsubseteq Y)\}$, S is not a closed set.*

Proof: For the sake of contradiction assume that S is a closed set. Then, after a round r when the state is $s \in S$, the chain remains in S forever. Given that $|Y| > n/2$ and the assumption that S is a closed set, at all rounds $r' > r$ we must have $cheaters(r') < n/2$. Then, given that $\forall i \in Z : WB_{\mathcal{Y}} \geq a_i + WC_{\mathcal{T}}$, from the transition function it can be seen that the probability of cheating of all workers in Z decreases in every round, independently of whether the master audits or not. But then, at some round $r' > r$, for all $i \in Z$, $p_{Ci}^{r'} = 0$ must hold. Then, $Z \subseteq Y$ at round r', showing that S is not closed. ■

Lemma 4.5. *Consider any set of workers $Z \subseteq W$ such that $\forall i \in Z : WB_{\mathcal{Y}} \geq a_i + WC_{\mathcal{T}}$ and $\forall j \notin Z : WB_{\mathcal{Y}} < a_j + WC_{\mathcal{T}}$. If $|Z| > n/2$ and $p_{\mathcal{A}} > 0$, then for any set of states $S = \{(p_{\mathcal{A}}, p_{C1}, \ldots, p_{Cn}) | \exists Y \subseteq W : (|Y| > n/2) \wedge (\forall w \in Y : p_{Cw} > 0)\}$, S is not a closed set.*

Proof: To prove this claim, it is enough to show that, after a round r when the state is $s \in S$, with some positive probability the chain moves out of S. Assume that, starting at some round $r' \geq r$, the master audits in all rounds in $[r', r'']$, for a suitable r''. Such assumption is valid because $p_{\mathcal{A}} > 0$. Then, given that $\forall i \in Z : WB_{\mathcal{Y}} \geq a_i + WC_{\mathcal{T}}$, from the transition function it can be seen that the probability of cheating of all workers in Z decreases in every round by an amount not smaller than $\alpha_w \min\{ WB_{\mathcal{Y}} - a_i - WC_{\mathcal{T}}, WP_{\mathcal{C}} + a_i\}$. But then, at some round $r'' > r$, for all $i \in Z$ we have $p_{Ci}^{r''} = 0$. Therefore, $|Y| < n/2$ at round r'' showing that S is not closed. ■

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. Its proof follows directly from Lemmas 4.3– 4.5.

Theorem 4.1. *Let $Z \subseteq W$ be any set of workers such that $|Z| > n/2$. If $p_{\mathcal{A}} > 0$ and for all $i \in W : \alpha_w(a_i + WP_{\mathcal{C}}) < 1$ 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 $WB_{\mathcal{Y}} \geq a_i + WC_{\mathcal{T}}$ for all $i \in Z$ in some set $Z \subseteq W$ such that $|Z| > n/2$.*

Remark 4.1. *From Algorithm 7 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 $p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$ and stays such forever.*

Convergence Time

Theorem 4.1 gives 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 the Markov chain takes 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 4.1, 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 $1/(\alpha_w \min\{ WB_{\mathcal{Y}} - a_i - WC_{\mathcal{T}}, WP_{\mathcal{C}} + a_i\})$ audits to enforce $p_C = 0$ for all covered workers $i \in C$. This gives the following upper bound on the convergence time:

Theorem 4.2. *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 4.2 may be too pessimistic for certain values of the parameters. The following theorem provides a tighter bound under certain conditions.

Theorem 4.3. *Let us define, for each worker i, $dec_i \triangleq \alpha_w \min\{WP_{\mathcal{C}} + a_i, WB_{\mathcal{Y}} - WC_{\mathcal{T}} - a_i\}$, $inc_i \triangleq \alpha_w \max\{WB_{\mathcal{Y}} - a_i, WC_{\mathcal{T}} + a_i\}$. 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 $0 < \varepsilon < 1 - \max_{i\in C}\{inc_i/(inc_i + dec_i)\}$, then the expected convergence time is $1/(\varepsilon(\min_{i\in C}\{dec_i\} + \max_{i\in C}\{inc_i\}))$.*

Proof: Let us define a potential function ϕ over the rounds as follows. Initially $\phi(0) = \max_{i\in C}\{p_{Ci}^0\}$. Then, for each round $r > 0$, $\phi(r) = \phi(r-1)$ if $\phi(r-1) = 0$. If $\phi(r-1) > 0$, then $\phi(r) = \max(0, \phi(r-1) - \min_{j\in C}\{dec_j\})$ if the master audits in round $r > 0$, and $\phi(r) = \phi(r-1) + \max_{j\in C}\{inc_j\}$ otherwise.

Consider any worker $i \in C$, and observe that, in the extreme cases, p_{Ci} decreases by $\min_{j\in C}\{dec_j\}$ when the master audits and increases by $\max_{j\in C}\{inc_j\}$ when the master does not audit. Also, once all workers in C have $p_{Ci} = 0$, this value does not change, since there is a majority of honest workers. Hence, it is clear that for all $r \geq 0$, $\phi(r) \leq \max_{i\in C}\{p_{Ci}^r\}$.

As a worst case, assume that $\phi(0) = 1$. We compute the expected number of rounds needed to get $\phi = 0$ 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 $1/(\varepsilon(\min_{i\in C}\{dec_i\} + \max_{i\in C}\{inc_i\}))$. ■

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

Corollary 4.1. *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 $\rho = 1/(\alpha_w WB_{\mathcal{Y}})$.*

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 4.6. *Let $p_{\mathcal{A}}^0 = p > 0$. 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 4.7. *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 4.8. *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 4.7 and $dec\_cheater \geq dec\_honest$, we have that

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

as desired. ■

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 4.9. *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 $p_{Ci}^r \leq \max\{0, p_{Ci}^{r-1} - dec\}$, independently of whether the master audits. Hence the proof. ■

Theorem 4.4. *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* $p_{\mathcal{A}}^0 = p > 0$*,* $dec\_cheater \geq dec\_honest$ *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 4.6, 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 4.8, $P^r \leq n/3$ with probability at least $1-\exp(-n/96)$. Then starting at round $r+1$, from Lemma 4.9, 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. ■

4.3.4 General Simulations

This subsection complements our analytical results with illustrative simulations. The graphical representation of the data obtained captures the tradeoffs between reliability and cost, a concept hard to view through the analysis. This is important as 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. The plots of Figure 4.4 are an exception and the plotted values represent only 1 execution of the implementation; the purpose is to illustrate the per round cost of the master.

Simulation Parameters

We choose sensible parameter values likely to be encountered in real applications; the choice of the parameters was influenced by statistics obtained from experiments contacted in SETI-like projects ([6, 50, 121]). 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

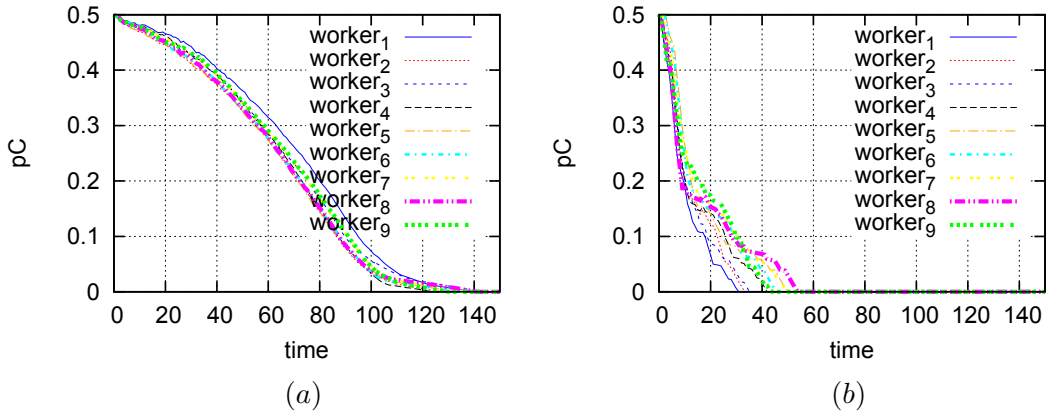


Figure 4.1: **Cheating probability for each worker as a function of time (number of rounds)** for parameters $p_C = p_{\mathcal{A}} = 0.5$, $WB_{\mathcal{Y}} = 1$, $WP_{\mathcal{C}} = 0$, $WC_{\mathcal{T}} = 0.1$ and $a_i = 0.1$. (a) $\alpha = 0.01$; (b) $\alpha = 0.1$.

typically each task is assigned to three workers [79]. So, in that context, nine workers seems an appropriate workforce. The initial cheating probability of each worker i is not known, therefore we have experimented with $p_{Ci} = 0.5$, as a reasonable assumption, and with $p_{Ci} = 1$ as an extreme case. Similarly, we have set $p_{\mathcal{A}} \in \{0.5, 1\}$ 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. Besides this intuition on the value of tolerance, we also carried out a set of experiments to understand the effect of this parameter, on which we will comment below.

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. In the simulations, unless otherwise stated, the master covers all workers and not some majority as assumed in the analysis (as a worst case scenario with respect to the master's cost).

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 $a_i \in \{0.01, 0.1\}$. 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 4.1 are satisfied and hence eventual convergence is reached. Finally, we consider the same learning rate for the master and the workers, i.e., $\alpha = \alpha_m = \alpha_w$. For practical reasons [128] it must be set to a small constant value, so we consider $\alpha \in \{0.1, 0.01\}$.

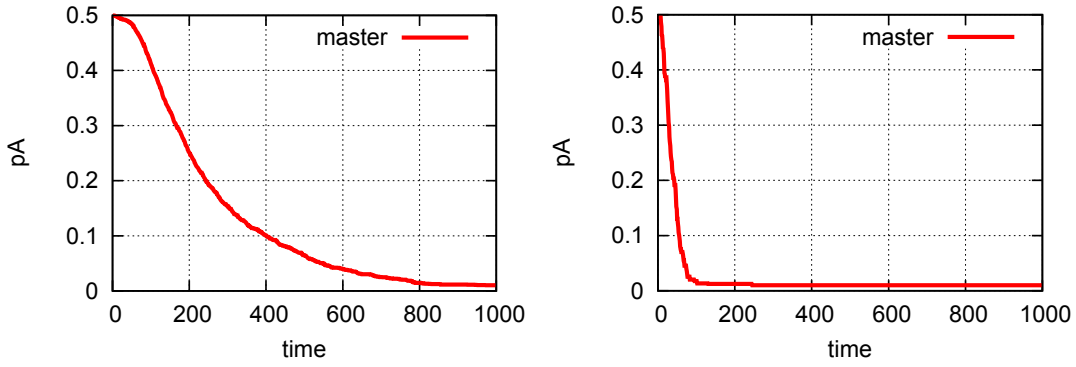


Figure 4.2: **Auditing probability for the master as a function of time (number of rounds)** for parameters $p_C = p_{\mathcal{A}} = 0.5$, $WB_{\mathcal{Y}} = 1$, $WP_{\mathcal{C}} = 0$, $WC_{\mathcal{T}} = 0.1$ and $a_i = 0.1$. (a) $\alpha = 0.01$; (b) $\alpha = 0.1$.

Convergence Time

Figure 4.1 shows that convergence can be reached very quickly, even without punishing cheaters and with small $WB_{\mathcal{Y}}$. 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 4.1 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 learning procedure will become unstable and p_C will bounce up and down without reaching convergence.) Similar conclusions can be drawn from Figure 4.2, 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 to deviate from the desirable behavior.

Effects of Punishment

Notice that in previous simulations only a positive reinforcement is applied to the workers (i.e., $WP_{\mathcal{C}} = 0$). Now, from Figure 4.3 we can notice that the larger the punishment we apply (i.e., $WP_{\mathcal{C}} \in \{1, 2\}$) the faster the convergence time is. In fact, we may conclude that applying only punishment is enough to have fast convergence. Comparing Figure 4.1(b) with Figure 4.3(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 reveals a trade-off between convergence time and the cost the master has for reaching faster convergence and maintaining it. Thus, the master could choose between different protocols estimating the cost of the auditing until it reaches convergence. But less auditing leads to larger convergence times. So it is not clear initially what is going to be optimal.

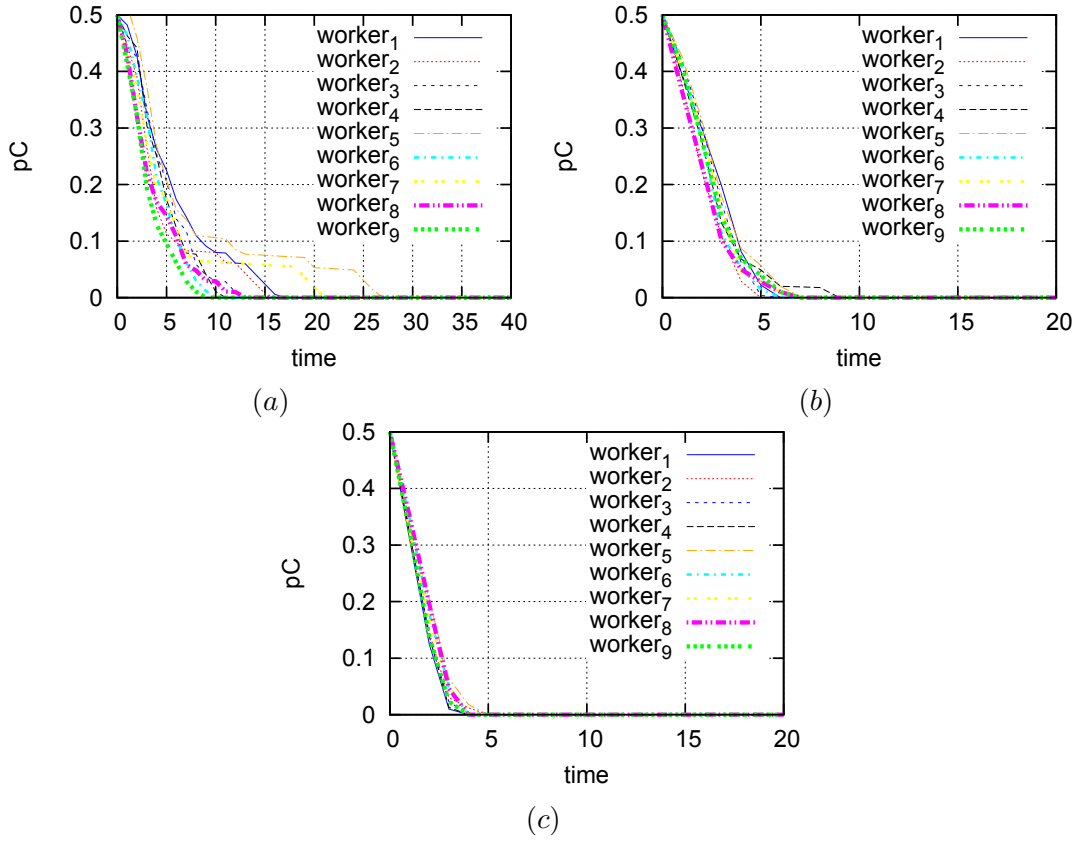


Figure 4.3: **Cheating probability for each worker as a function of time (number of rounds)**for parameters $p_C = p_{\mathcal{A}} = 0.5$, $\alpha = 0.1$, $a_i = 0.1$, $WB_{\mathcal{Y}} = 2$ and $WC_{\mathcal{T}} = 0.1$. (a) $WP_{\mathcal{C}} = 0$; (b) $WP_{\mathcal{C}} = 1$; (c) $WP_{\mathcal{C}} = 2$.

Master's Cost

Recall from the simulation parameters that only the cost of auditing and the workers' payment are non-zero, and that the master covers all workers (instead of some majority as in the analysis). We contrasted first a worst case scenario where workers initially cheat with probability $p_C = 1$, against the case where, given the lack of knowledge about worker's behavior, we assume that initially $p_C = p_A = 0.5$.

The first conclusion we can draw from our simulations is that, even in this unfavorable situation, eventual convergence is still achieved. However, during the process the master's auditing probability reaches 1 for the system to converge. Of course, this has a direct impact on the cost of convergence, but also on the convergence time. Denote by $p_{\mathcal{A}}(0)$ the master's initial auditing probability. Interestingly, from Figures 4.4(a2), (a3), (b2), and (b3) we observe that, when $p_{\mathcal{A}}(0) = 1$, the convergence time decreases by 9%, and that even $p_{\mathcal{A}}$ converges to its minimum in 25% less time yielding also a cost reduction. In fact, during the first 10 rounds of evolution, for $p_{\mathcal{A}}(0) = 0.5$ the aggregate cost for the master is 56% smaller (Figure 4.4(a1)) than when $p_{\mathcal{A}}(0) = 1$ (Figure 4.4(b1)). However, in the subsequent interval between $p_{\mathcal{A}}^{min} < p_{\mathcal{A}} < 1$ the sit-

uation is reversed, and the master's cost for $p_{\mathcal{A}}(0) = 1$ is 19% smaller than the case $p_{\mathcal{A}}(0) = 0.5$. These results show that using $p_{\mathcal{A}}(0) = 1$ does not necessarily increase cost as the intuition might suggest. After convergence is achieved, Figures 4.4(a1), (b1), and (c1) show that, once the master's auditing probability has reached its minimum value, the master audits roughly once every hundred rounds, which is expected given that $p_{\mathcal{A}}^{min} = 0.01$. This behavior is observed independently of the initial auditing probability, which is also expected.

Another surprising result, arising from Figures 4.4(b3) and (c3), is that having workers with larger aspiration values makes the convergence time decrease by more than a half. The reason being that the master initially audits with probability one and all workers cheat, so a larger aspiration causes the workers' cheating probability to drop at a higher rate. This, in turn, feeds back to the master's auditing probability and cost, making them decrease faster than the case where workers have a smaller aspiration.

We have also examined the case where only a majority of workers is covered. Specifically, we have run analogous simulations to the ones depicted in Figure 4.4, but now covering only 5 out of the 9 workers. A first, interesting observation, is that the convergence time for the covered workers is not affected. An even more interesting observation is that the master's cost, until $p_{\mathcal{A}}^{min}$ was reached, is greater than the case of all-covered workers. This is due to the slower rate at which $p_{\mathcal{A}}$ reaches its minimum value. Of course, after this point, the master's cost is smaller since it rewards fewer workers.

Tolerance Value

Finally, we have also considered the effect of tolerance for achieving eventual convergence using three different initial $p_C \in \{0.3, 0.5, 1\}$. Choosing 5000 iterations as a value large enough to illustrate the problems of convergence for high tolerance (values of the same order of magnitude or larger give basically the same results), we have found that convergence will always be reached for the lower values of p_C. This is due to the fact that with "almost" honest workers, a majority of them will always compute the answer and will force punishment to the minority of cheaters. However, when the initial p_C equals one and there is no punishment ($WP_{\mathcal{C}} = 0$), the master must respond to a percentage of cheating workers (due to tolerance) to obtain eventual convergence, and as a consequence convergence is not achieved for large tolerance values (for this specific set of parameters, when $\tau > 0.9$). Interestingly, with non-zero punishment ($WP_{\mathcal{C}} = 1$), the master can tolerate all workers cheating and still achieve convergence. Therefore, our simulated examples establish that, based on the rest of the parameter values, the master can appropriately set values for tolerance τ, (its own) learning rate α, punishment $WP_{\mathcal{C}}$ and perhaps $WB_{\mathcal{Y}}$ in such a way that convergence can be swiftly obtained.

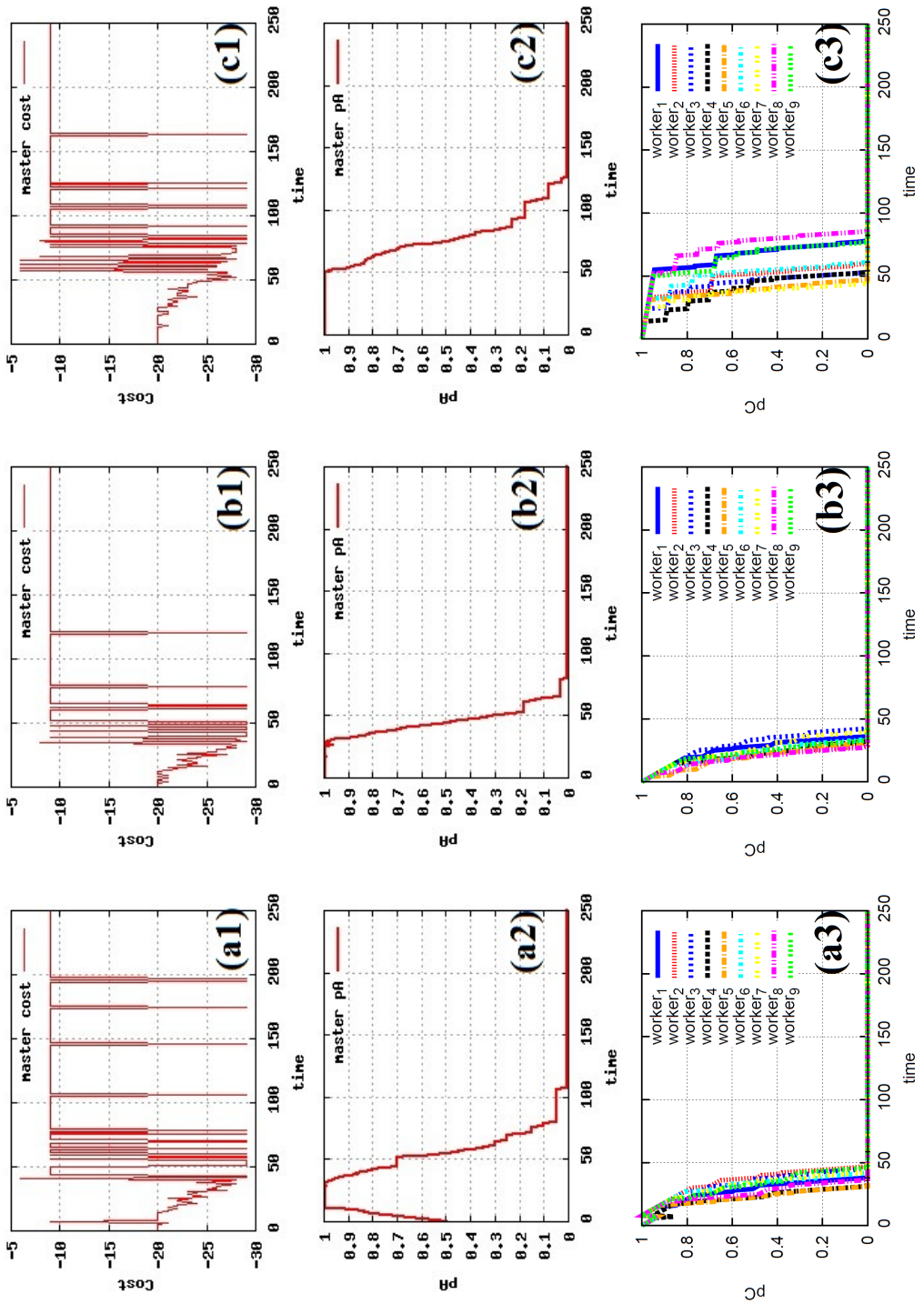


Figure 4.4: **Top panel, master's cost as a function of time. Middle panel, master's auditing probability as a function of time. Bottom panel, worker's cheating probability as a function of time.** Parameters in all panels, $p_C = 1$, $WC_{\mathcal{T}} = 0.1$, $WP_{\mathcal{C}} = 0$ and $\alpha = 0.1$. Left panel, $p_{\mathcal{A}} = 0.5$, $a_i = 0.1$. Middle panel $p_{\mathcal{A}} = 1$, $a_i = 0.1$. Right panel $p_{\mathcal{A}} = 1$, $a_i = 0.01$.

4.3.5 Large Scale Simulations

For the sake of experimentation, in this subsection we test the designed mechanism under extreme parameter values. The reason is twofold, first of all, we want to observe the time the system needs to reach eventual correctness when the number of participating workers is a large. At a second level, we want to examine better the tolerance parameter, for which we don't have much information from the analysis.

Parameters

We make the assumption that workers are rational and have no information about the master's initial probability of auditing. The same applies for the master's initial probability of auditing, hence, we have taken $p_{\mathcal{A}} = 0.5$ as the initial value. The minimum probability of auditing will be set to be $p_{\mathcal{A}}^{min} = 0.01$. We set the initial cheating probability of each worker i to $p_{Ci} = 0.1$ as a worst case condition in terms of time to reach eventual correctness. As for the tolerance, this will be one of our parameters of interest so we will sweep a range of values in what follows.

Let us now proceed with the payoffs for the workers. We will take the worker's reward for correct answers, $WB_{\mathcal{Y}} = 1$, by way of normalization and, within that framework, we will choose $WP_{\mathcal{C}} = \{0, 1\}$ 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. As we will see, those values of punishment are enough to understand the system behavior, and, as for the computing cost, we believe that it is reasonable to assume that it is not too large compared with the reward. Regarding the aspiration level, as we have already said, this is a parameter defined by the workers in an idiosyncratic manner; for simplicity, in these simulations we fix a uniform aspiration level, $a_i = \{0.01, 0.1\}$. 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 [128] (called step-size there), for practical reasons can be set to a small constant value; so we consider $\alpha = 0.1$.

Results

In this subsection, as we mentioned before, we are mainly interested on τ, the parameter expressing the master's tolerance to cheating. Fig. 4.5 collects our results on the influence of tolerance on convergence, showing the percentage of runs that converged to eventual correctness out of a batch of 50 realizations with a runtime of 1000 rounds for every tolerance value and four numbers of workers. The first conclusion we can draw from these plots is that for the case without punishment and with the lowest aspiration level (top left panel) all realizations yielded eventual correctness for tolerances as large as $\tau = 0.6$. That is, the system remained well behaved (in the sense of providing correct answers) even if the master does not react to a majority of defectors. For higher tolerances the system behavior worsens. Lack of convergence within our simulation time becomes more frequent. For those realizations that do not converge, we cannot be sure that they will reach eventual correctness at some later time. Therefore, the percentage of realizations

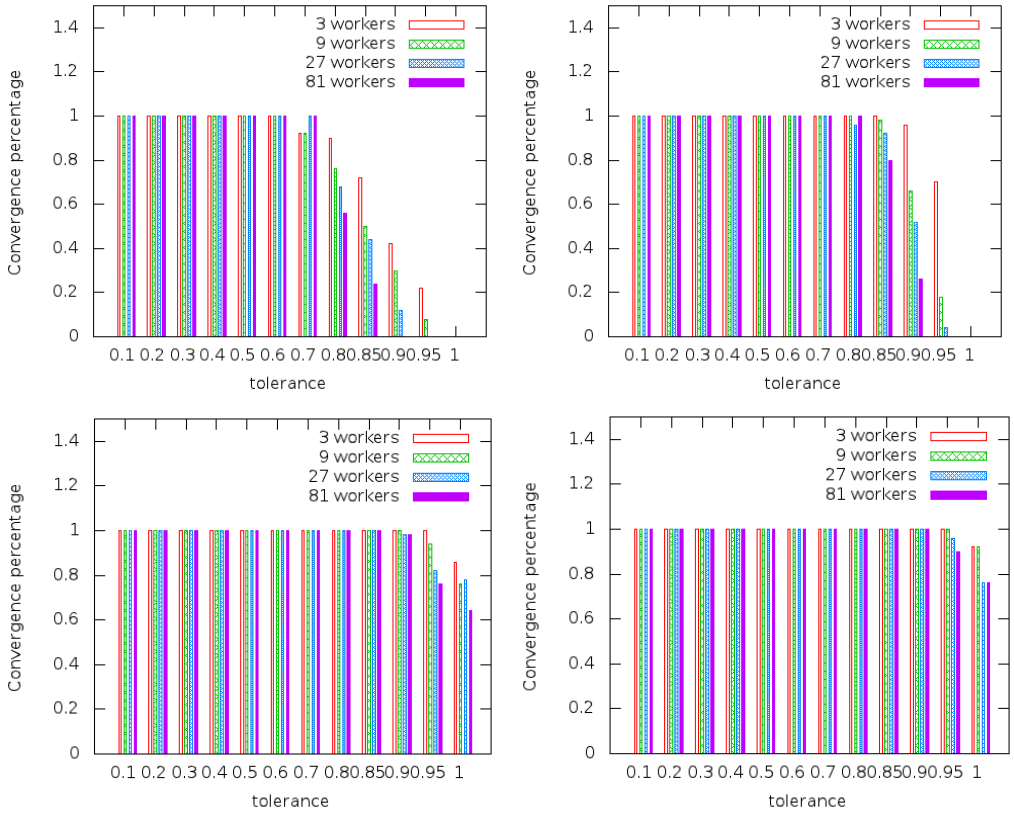


Figure 4.5: **Convergence percentage of 50 realizations in 1000 rounds.** Initial condition in all cases was $p_c = 1$ for all workers, $p_A = 0.5$. Parameters in all panels, $WC_{\mathcal{T}} = 0.1$, $\alpha = 0.1$, $p_{\mathcal{A}}^{min} = 0.01$. Left panels: $a_i = 0.01$, right panels: $a_i = 0.1$. Top panels, $WP_{\mathcal{C}} = 0$, bottom panels, $WP_{\mathcal{C}} = 1$.

that converged would be an estimate for a lower bound to the probability of reaching eventual correctness. As we can see from the rest of the panels, increasing the aspiration level to $a_i = 0.1$ takes the maximum allowed tolerance that still has eventual correctness in every realization up to $\tau = 0.7$, whereas introducing punishment for defectors is also an improvement factor, leading to threshold tolerances of $\tau = 0.85$ for the low aspiration level and even to $\tau = 0.9$ for the largest one. A value such as $\tau = 0.9$ is certainly very large; the fact that the system ends up providing correct answers even when the master only updates her auditing probability in extreme cases shows the robustness of the design.

Interestingly, Fig. 4.5 also shows the relevance of the number of workers in the system performance. It appears from the plots that when the master takes her tolerance beyond the convergence threshold the probability to reach eventual correctness decreases if the number of workers is increased. Except for the results at the threshold ($\tau = 0.7$) in the case with no punishment and low aspiration (top left plot), increasing the number of workers always lead to worse results in terms of convergence for any value of the tolerance. We believe that this phenomenon arises because increasing the number of workers and keeping everything else the same, it is more probable that one of the workers defects. In this large tolerance region, the behavior would not change the master's auditing probability and, given that there is no punishment, reinforcement learning would not act so strongly on the corresponding worker, making it more difficult for the system to converge.

Further information on the dependence of the system on the number of workers is provided by Fig. 4.6, where we study one of the cases of Fig. 4.5 in more detail. For the lowest aspiration value, $a_i = 0.01$, we see that for tolerances $\tau = 0.75$ and higher, the convergence percentage decreases with the number of workers. The system sizes we are able to study at this time are not enough to ascertain whether it eventually reaches zero for sufficiently large number of workers, which would indicate an abrupt transition in the performance of the algorithm upon changing the tolerance. Increasing the aspiration level we observe, in agreement with our earlier remarks, that the system performs much better in terms of higher convergence rates, and also that the transition to the non-performing system seems somewhat more abrupt, going from full convergence (i.e., convergence percentage equal to 1) for 2217 workers at $\tau = 0.8$ to a very low value for $\tau = 0.9$. However, specific claims about the nature of this transition cannot be made without studying the whole parameter space. What we can certainly conclude is that the degrading of the performance becomes worse for larger number of workers, as hinted from previous plots, and that for tolerance values up to $\tau = 0.7$ the system always provides correct answers. We thus confirm the robustness of our design as far as the need for the master to update her auditing probability is concerned, as she can obtain good results just keeping the same value (and hence auditing less often and saving costs) unless an inordinate number of workers defect. In addition, having workers with high aspirations allows for an even lower rate of increase of the auditing probability.

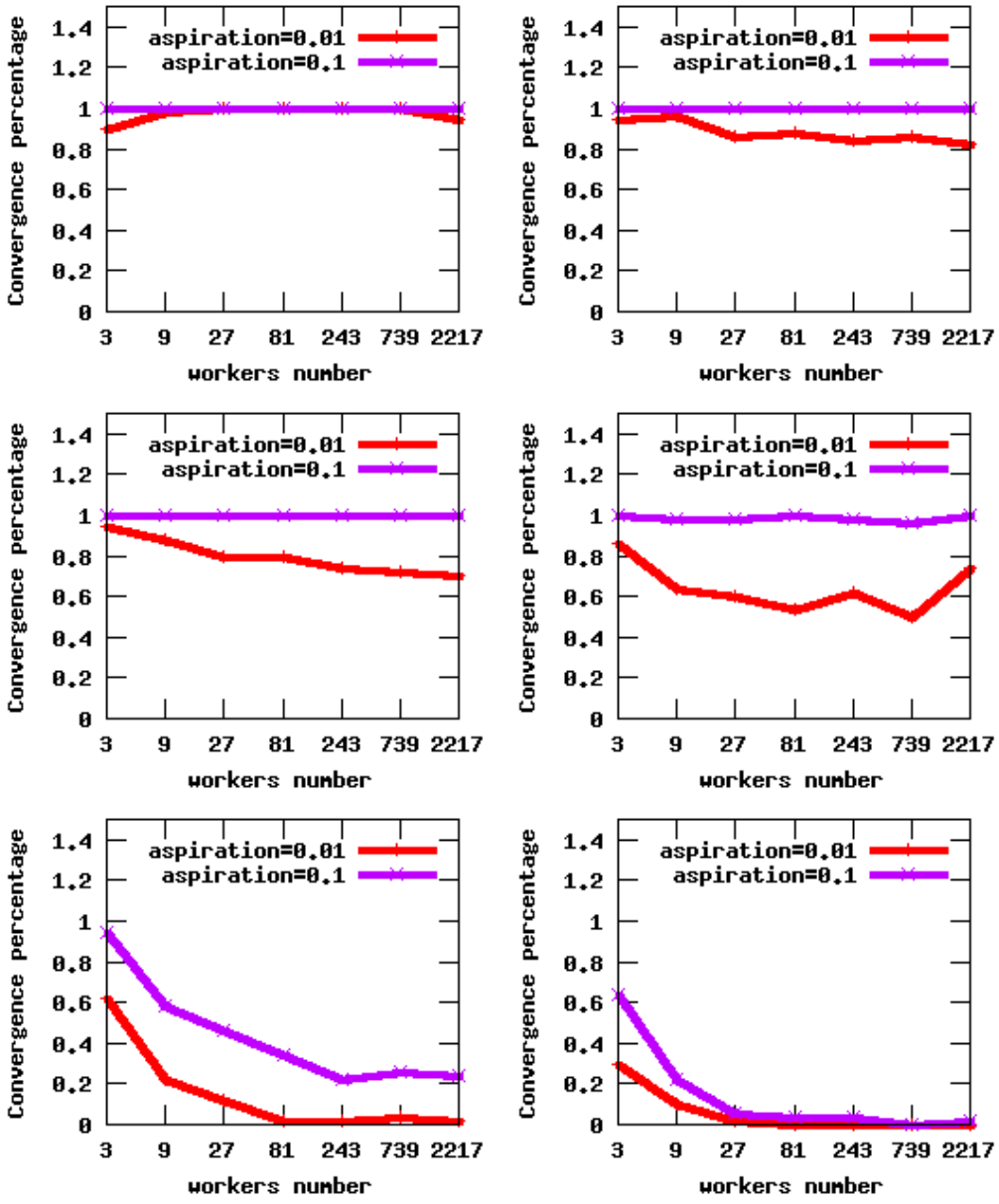


Figure 4.6: **Convergence percentage of 50 realizations in 1000 rounds as a function of the number of workers** for $WP_{\mathcal{C}} = 0$, $WC_{\mathcal{T}} = 0.1$, $\alpha = 0.1$, $p_{\mathcal{A}}^{min} = 0.01$, with either $a_i = 0.01$ (red lines) or $a_i = 0.1$ (purple lines). From top to bottom and from left to right, tolerance values are $\tau = 0.7, 0.75, 0.775, 0.8, 0.9, 0.95$. Lines are only a guide to the eye.

Discussion

Looking into simulation scenarios where a large number of workers takes place has allowed us to look in detail into the tolerance paremeter, something that our analytical results do not provide much information.

As we have seen, the system is able to perform correctly and provide the right answer for high tolerance values, implying that the master needs only to react to large majorities of defectors or cheaters. In fact, using punishment may push the tolerance levels for which convergence to eventual correctness is always guaranteed up to values as large as $\tau = 0.85$, meaning that the master would only need to increase her auditing probability when almost all workers are cheating. This is a very good result not only in terms of the convergence of the system but also as regards the cost, because lower auditing probabilities means less actual audits and therefore less incurred costs by the master. On the other hand, the tolerance behavior above the convergence threshold depends on the number of workers participating in the crowd computing process. As we have seen, the performance of the system decreases with the number of workers for both aspiration values studied, the decrease being slower (but starting at lower tolerances) for the lowest aspiration level. This observation points to the need of balancing the requirements of the task to be carried out in terms of number of workers involved with the reliability of the process (convergence percentage). Keeping everything else constant, increasing the number of workers may lead to a serious degrading of the results, so if more workers are required it is probably necessary to lower the tolerance to keep the worker population in check.

4.4 Presence of Malicious, Altruistic and Rational Workers

4.4.1 Introduction

Like in the previous section, we are looking into the problem of designing mechanisms for establishing reliability in task computing systems over the Internet. Moving ahead our original assumption, that workers are rational, we consider that workers can be categorized in three generic types: *altruistic*, *malicious* and *rational*. Altruistic workers that always return the correct result, malicious workers that always return an incorrect result, and rational workers that decide to reply or not truthfully depending on what increases their benefit.

Studying individually the case where only rational workers are present has allowed us to design a reinforcement learning mechanism to induce a correct behavior to rational workers, without having to consider other behaviors from the workers. In this section, we move forward and supported by the literature we assume that besides rational workers also malicious and altruistic workers can exist. Thus adding to the existing mechanism and taking advantage of the repeated interaction of the master with the workers we complement the mechanism with four reputation schemes that cope with malice. The goal of the mechanism is to reach a state of *eventual correctness*, that is, a stable state of the system in which the master always obtains the correct task results. Analysis of the system gives provable guarantees under which truthful behavior can be ensured by modelling the system as a Markov chain. Finally, we observe the behavior of the mechanism through simulations that use realistic system parameters values. Simulations not only agree with the analysis but also reveal interesting trade-offs between various metrics and parameters. The correlation among cost and convergence time to a truthful behavior is shown and the four reputation schemes are assessed against the tolerance to cheaters.

Our Contributions

In this section, we aim at establishing a reliable Internet-based task computing system. Entities follow the master-worker model where the master assigns tasks to a fixed set of workers in an online fashion. (Prediction mechanisms such as the one in [87] can be used to establish the availability of a set of workers for a relatively long period of time.) Workers on the other hand are active, aware of their repeated interaction with the master and willing to reply. They are categorized into three types: (1) altruistic, (2) malicious and (3) rationals. The good behavior of the rational workers is reinforced through an incentives mechanism, while the malicious workers are being identified through a number of reputation schemes proposed. The goal of the system is to achieve a stable state where the master will always receive the correct task reply with the minimum cost to the master (i.e. auditing), from there forth. In detail, our contributions are as follows.

- We design such an algorithmic mechanism that uses reinforcement learning (through reward and optional punishment) to induce a correct behavior to rational workers while coping with malice using *reputation*, see Subsection 4.4.3.

- We consider a centralized reputation scheme controlled by the master that may use four different reputation metrics to calculate each worker's reputation. The first (reputation type Linear) is adopted from [127] and it is a simple approach for calculating reputation. The second reputation type (called Exponential) , which we introduce, allows for a more drastic change of reputation simply because the mathematical function used changes faster. The third reputation type is inspired by Berkeley Open-source software for volunteer computing (BOINC) current reputation scheme [11], thus we call it Boinc. Finally, the fourth reputation scheme [11] is inspired by the previously used reputation scheme [10] of BOINC (called Legacy Boinc) and it uses an indirect way of calculating reputation through an error rate. All four reputation schemes are presented in Subsection 4.4.2.

- We analyze, in Subsection 4.4.4 our reputation-based mechanism modeling it as a Markov chain and we identify conditions under which truthful behavior can be ensured. We analytically prove that by using the reputation type Exponential (i.e. the one we introduce) reliable computation is eventually achieved.

- Simulation results in Subsection 4.4.5, obtained using parameter values extracted by BOINC-operated applications (such as [6, 50]), reveal interesting trade-offs between various metrics and parameters, such as cost, time of convergence to a truthful behavior, tolerance to cheaters and the type of reputation metric employed. Simulations also reveal better performance of our reputation type (Exponential) in several realistic cases.

4.4.2 Model

In this subsection we complement the general model presented in Section 4.2. Remember that in Section 4.2 we presented the concepts of auditing, payoffs, rewards and aspiration. In this subsection we complement the model by formally defining the four reputation types that are used by the mechanism.

Reputation

The reputation of each worker is measured by the master; a centralized reputation mechanism is used. 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.

In this work, we consider *four* reputation metrics. The first one is analogous to a reputation metric used in [127] and we call it *Linear* in this work. Reputation *Boinc* is inspired by the BOINC adaptive replication metric currently in use [11], while reputation metric *Legacy Boinc* is inspired

by the previously used version of BOINC's adaptive replication metric [10]. We present the performance of our system under both BOINC reputation metrics as an opportunity to compare and contrast these two schemes within our framework. Finally, the last reputation metric we consider is reputation *Exponential* that is not influenced by any other reputation type, and as we show in Section 4.4.4 it possesses beneficial properties. In all types, the reputation of a worker is determined based on the number of times it was found truthful. Hence, the master may update the reputation of the workers only when it audits. We denote by $aud(r)$ the number of rounds the master audited up to round r, and by $v_i(r)$ we refer to the number of auditing rounds in which worker i was found truthful up to round r. Moreover, we define $streak_i(r)$ as the number of rounds $\leq r$ in which worker i was audited, and replied correctly after the latest round in which it was audited and caught cheating. We let $\rho_i(r)$ denote the *reputation* of worker i after round r, and for a given set of workers $Y \subseteq W$ we let $\rho_Y(r) = \sum_{i \in Y} \rho_i(r)$ be the aggregated reputation of the workers in Y, by aggregating we refer to summing the reputation values. Then, the reputation types we consider are detailed in Figure 4.7.

Linear: $\rho_i(r) = (v_i(r) + 1)/(aud(r) + 2)$.

Exponential: $\rho_i(r) = \varepsilon^{aud(r) - v_i(r)}$, for $\varepsilon \in (0, 1)$, when $aud(r) > 0$, and $\rho_i(r) = 1/2$, otherwise.

Legacy Boinc: Here we define $\beta_i(r)$ as the error rate of worker i at round r. Reputation for this type is calculated as follows:

$$\beta(r) = \begin{cases} 0.1, & \text{if } r = 0. \\ 0.95\beta(r-1), & \text{if } r > 0 \text{ and worker is truthful in round } r. \\ \beta(r-1) + 0.1, & \text{otherwise.} \end{cases}$$

$$\rho(r) = \begin{cases} 0.5, & \text{if } r = 0. \\ 0, & \text{if } \beta(r) > 0.05. \\ 1 - \sqrt{\frac{\beta(r)}{0.05}}, & \text{otherwise.} \end{cases}$$

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

Figure 4.7: **Reputation types.**

These four reputation types satisfy the following two natural properties:

A. If a worker is honest when the master audits, the reputation of the worker cannot decrease.

B. If a worker cheats when the master audits, the reputation of the worker cannot increase.

This claim is proven below in Lemmas 4.10 and 4.11.

Lemma 4.10. *Natural Property A holds for reputation type Linear, Exponential, Legacy Boinc and Boinc.*

Proof: We present separately the proof for each reputation type.
Linear: Assume that at state s_r worker i has reputation $\rho_i(r) = \frac{v_i(r)+1}{aud(r)+2}$ and in the next state the master audits and the worker is honest then reputation becomes $\rho_i(r+1) = \frac{v_i(r)+2}{aud(r)+3}$. Since $\frac{v_i(r)+1}{aud(r)+2} \leq \frac{v_i(r)+2}{aud(r)+3}$ the lemma holds for reputation type Linear.
Exponential: Assume that at state s_r worker i has reputation $\rho_i(r) = \varepsilon^{aud(r)-v_i(r)}$ and in the next state the master audits and the worker is honest then reputation becomes $\rho_i(r+1) = \varepsilon^{aud(r)-v_i(r)}$, then the lemma trivially holds for reputation type Exponential.
Legacy Boinc: At state s_r worker i can have reputation $\rho_i(r) \geq 0$ and in the next state the master audits and the worker is honest then $\beta_i(r) < 0.05$ and $\rho_i(r+1) \geq 0$. If in state s_r $\rho_i(r) = 0$ then the natural property holds. If in state s_r, $\rho_i(r) = 1 - \sqrt{\frac{\beta_i(r)}{0.05}}$ then in the next state $\rho_i(r+1) = 1 - \sqrt{\frac{\beta_i(r+1)}{0.05}}$. The property still holds since $\beta_i(r) < \beta_i(r+1)$ and thus the claim is proved for reputation type Legacy Boinc.
Boinc: Three possible cases exist: 1)Assume that at state s_r worker i has reputation $\rho_i(r) = 0$ and $streak_i(r) < 9$ and in the next state the worker is honest, then reputation becomes $\rho_i(r+1) = 0$. Since $\rho_i(r) = \rho_i(r+1)$ the lemma holds for this case. 2)Assume that at state s_r worker i has reputation $\rho_i(r) = 0$ and $streak_i(r) = 9$ and in the next state the worker is honest, then reputation becomes $\rho_i(r+1) = \frac{9}{10}$. Since $0 \leq \frac{9}{10}$ the lemma holds for this case too. 3)Finally, assume that at state s_r worker i has reputation $\rho_i(r) = 1 - \frac{1}{streak_i(r)}$ and $streak_i(r) \geq 10$ and in the next state the worker is honest, then reputation becomes $\rho_i(r+1) = 1 - \frac{1}{streak_i(r+1)}$. Since $streak_i(r) < streak_(r+1)$ then $1 - \frac{1}{streak_i(r)} \leq 1 - \frac{1}{streak_i(r+1)}$ and the lemma holds for this case. Thus, the lemma for reputation type Boinc holds since it is true for all three possible cases. ■

Lemma 4.11. *Natural Property B holds for reputation type Linear, Exponential, Legacy Boinc and Boinc.*

Proof: We present separately the proof for each reputation type.
Linear: Assume that at state s_r worker i has reputation $\rho_i(r) = \frac{v_i(r)+1}{aud(r)+2}$ and in the next state the master audits and the worker cheats then reputation becomes $\rho_i(r+1) = \frac{v_i(r)+1}{aud(r)+3}$. Since $\frac{v_i(r)+1}{aud(r)+2} \geq \frac{v_i(r)+1}{aud(r)+3}$ the lemma holds.
Exponential: Assume that at state s_r worker i has reputation $\rho_i(r) = \varepsilon^{aud(r)-v_i(r)}$ and in the next state the master audits and the worker cheats then reputation becomes $\rho_i(r+1) = \varepsilon^{aud(r)+1-v_i(r)}$. Since $\varepsilon \in (0,1)$ then $\varepsilon^{aud(r)-v_i(r)} \geq \varepsilon^{aud(r)+1-v_i(r)}$ and the lemma holds.
Legacy Boinc: At state s_r worker i can have reputation $\rho_i(r) \geq 0$ and in the next state the master audits and the worker cheats then $\beta_i(r+1) > 0.05$ and $\rho_i(r+1) = 0$ and the claim holds.
Boinc: Two possible cases exist: 1)Assume that at state s_r worker i has reputation $\rho_i(r) = 0$ and

$streak_i(r) < 10$ and in the next state the master audits and the worker cheats, then $streak_i(r+1) = 0$ and reputation becomes $\rho_i(r+1) = 0$. Since $\rho_i(r) = \rho_i(r+1)$ the lemma holds for this case. 2)Assume that at state s_r worker i has reputation $\rho_i(r) = 1 - \frac{1}{streak_i(r)}$ and $streak_i(r) \geq 10$ and in the next state the master audits and the worker cheats. Then $streak_i(r+1) = 0$ and the reputation becomes $\rho_i(r+1) = 0$. Since, $1 - \frac{1}{streak_i(r)} > 0$ the lemma hold also for this case. Thus, the lemma holds in general. ■

In each round, when the master *does not audit*, the result is obtained from the *weighted majority* as follows. Consider a round r. Let $F(r)$ denote the subset of workers that returned an incorrect result, i.e., the rational workers who chose to cheat plus the malicious ones; recall that we assume as a worst case that all cheaters return the same value. Then, $W \setminus F(r)$ is the subset of workers that returned the correct value, i.e., the rational workers who chose to be truthful plus the altruistic ones. Then, if $\rho_{W \setminus F(r)}(r) > \rho_{F(r)}(r)$, the master will accept the correct value, otherwise it will accept an incorrect value. The mechanism, presented in the next subsection, employs auditing and appropriate incentives so that rational workers become truthful and have a reputation that is higher than that of the malicious workers.

4.4.3 Reputation-based Mechanism

We present now 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 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 possibly safe approach is to initially set $p_{\mathcal{A}} = 0.5$. The master also chooses the reputation type to use.

After that, at each round, the master sends a task to all workers and, when all answers are received, the master 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, and continues to the next round with the same probability of auditing and the same reputation values for each worker. In the case the answers are audited, the value $p_{\mathcal{A}}$ of the next round is reinforced (i.e., modified according to the accumulated reputation of the cheaters) and the reputations of the workers are updated based on their responses. Then, the master rewards/penalizes the workers appropriately. Specifically, if the master audits and a worker i is a cheater (i.e., $i \in F$), then $\Pi_i = -WP_{\mathcal{C}}$; if i is honest, then $\Pi_i = WB_{\mathcal{Y}}$. If the master does not audit, and i returns the value of the weighted majority (i.e., $i \in W_m$), then $\Pi_i = WB_{\mathcal{Y}}$, otherwise $\Pi_i = 0$.

We include a threshold, denoted by τ, that represents the master's *tolerance* to cheating (typ-

Algorithm 9 Master's Algorithm

$p_{\mathcal{A}} \leftarrow x$, *where* $x \in [p_{\mathcal{A}}^{min}, 1]$
$aud = 0$
// initially all workers have the same reputation
$\forall i \in W : v_i = 0; \beta_i = 0.1; \rho_i = 0.5; streak_i = 0$
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**
      // weighted majority, coin flip in case of a tie
      accept *the value returned by workers in* $W_m \subseteq W$,
          where $\rho_{W_m} > \rho_{W \setminus W_m}$
    else *// the master audits*
      $aud \leftarrow aud + 1$
      Let $F \subseteq W$ *be the set of workers that cheated.*
      $\forall i \in W$:
        // honest workers
        if $i \notin F$ **then** $v_i \leftarrow v_i + 1$ *or* $\beta_i \leftarrow \beta_i \cdot 0.95$ *or* $streak_i \leftarrow streak_i + 1$
        // cheater workers
        else $v_i \leftarrow v_i$ *or* $\beta_i \leftarrow \beta_i + 0.1$ *or* $streak_i \leftarrow 0$
        update *reputation* ρ_i *of worker* i *as defined*
            by reputation type used
      if $\rho_W = 0$ **then** $p_{\mathcal{A}} \leftarrow \min\{1, p_{\mathcal{A}} + \alpha_m\}$ **else**
        $p_{\mathcal{A}} \leftarrow \min\{1, \max\{p_{\mathcal{A}}^{min}, p_{\mathcal{A}} + \alpha_m(\frac{\rho_F}{\rho_W} - \tau)\}\}$
    $\forall i \in W$: **return** *payoff* Π_i *to worker* i

Algorithm 10 Algorithm for Rational Worker i

$p_{Ci} \leftarrow y$, *where* $y \in [0, 1]$
for $r \leftarrow 1$ **to** ∞ **do**
  receive *a task* T *from the master*
  $S_i \leftarrow -1$ *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
  if $S_i = 1$ **then**
    $\Pi_i \leftarrow \Pi_i - WC_{\mathcal{T}}$
  $p_{Ci} \leftarrow \max\{0, \min\{1, p_{Ci} - \alpha_w(\Pi_i - a_i)S_i\}\}$

ically, 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. 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 $p_{\mathcal{A}}$.) A pseudocode of the algorithm described is given as Algorithm 9.

Workers' Algorithm

This algorithm is run only by rational workers (recall that altruistic and malicious workers have a predefined behavior). The execution of the algorithm begins with each rational worker i deciding an initial probability of cheating p_{Ci}. In each round, each worker receives a task from the master and, with probability $1 - p_{Ci}$ computes the task and replies to the master with the correct answer. Otherwise, it fabricates an answer, and sends the incorrect response to the master. We use a flag S_i to model the stochastic decision of a worker i to cheat or not. After receiving its payoff, each worker i changes its p_{Ci} according to payoff Π_i, the chosen strategy S_i, and its aspiration a_i.

The workers have a *learning rate* α_w. In this work, we assume that all workers have the same learning rate, that is, they learn in the same manner (see also the discussion in [128]; the learning rate is called step-size there); note that our analysis can be adjusted to accommodate also workers with different learning rates.

4.4.4 Analysis

We now analyze the reputation-based mechanism. We model the evolution of the mechanism as a Markov Chain, and then discuss the necessary and sufficient conditions for achieving eventual correctness. Modeling a reputation-based mechanism as a Markov Chain is more involved than the previous model that does not consider reputation (see Subsection 4.3.3).

The Markov Chain

Let the state of the Markov chain be given by a vector s. The components of s are: for the master, the probability of auditing $p_{\mathcal{A}}$ and the number of audits before state s, denoted as aud; and for each *rational* worker i, the probability of cheating p_{Ci}, the number of *validations* (i.e., the worker was honest when the master audited) before state s, denoted as v_i, the error rate β_i and $streak_i$ (the number of consecutive times a workers was found honest since the last time she cheated). To refer to any component x of vector s we use $x(s)$. Then,

$$s = \big\langle p_{\mathcal{A}}(s), aud(s), p_{C1}(s), p_{C2}(s), \ldots, p_{Cn}(s), v_1(s), v_2(s), \ldots, v_n(s),$$
$$\beta_1(s), \beta_2(s), \ldots, \beta_n(s), streak_1(s), streak_2(s), \ldots, streak_n(s)\big\rangle.$$

In order to specify the transition function, we consider the execution of the protocol divided in rounds. In each round, probabilities and *counts* (i.e. numbers of validations, audits, error

rate and streak) are updated by the mechanism as defined in Algorithms 9 and 10. The state at the end of round r is denoted as s_r. Abusing the notation, we will use $x(r)$ instead of $x(s_r)$ to denote component x of vector s_r. The workers' decisions, the workers' error rate, the number of cheaters, and the payoffs of each round $r > 0$ are the stochastic outcome of the probabilities and counts at the end of round $r-1$. We specify the transition from s_{r-1} to s_r by the actions taken by the master and the workers during round r.

In the definition of the transition function that follows, the probabilities are limited to $p_{\mathcal{A}}(s) \in [p_{\mathcal{A}}^{min}, 1]$ and for each rational worker i to $p_{Ci}(s) \in [0,1]$, for any state s. The initial state s_0 is arbitrary but restricted to the same limitations. Let $P_F(r)$ be the probability that the set of cheaters in round r is exactly $F \subseteq W$. (That is, $P_F(r) = \prod_{j \in F} p_{Cj}(r-1) \prod_{k \notin F}(1-p_{Ck}(r-1))$.) Then, the transition from state s_{r-1} to s_r is as follows.

- Malicious workers always have $p_C = 1$ and altruistic workers always have $p_C = 0$.

- With probability $p_{\mathcal{A}}(r-1) \cdot P_F(r)$, the master audits when the set of cheaters is F. Then, according to Algorithms 9 and 10, the new state is as follows.

  For the master: $aud(r) = aud(r-1) + 1$ and, if $\rho_W(r) > 0$ then $p_{\mathcal{A}}(r) = p_{\mathcal{A}}(r-1) + \alpha_m\,(\rho_F(r)/\rho_W(r) - \tau)$ and $p_{\mathcal{A}}(r) = \min\{1, p_{\mathcal{A}} + \alpha_m\}$ otherwise.

  (1) For each worker $i \in F$: $v_i(r) = v_i(r-1)$, $\beta_i(r) = \beta_i(r-1)+0.1$ and $streak_i(r) = 0$ and, if i is rational, then $p_{Ci}(r) = p_{Ci}(r-1) - \alpha_w(a_i + WP_{\mathcal{C}})$.

  (2) For each worker $i \notin F$: $v_i(r) = v_i(r-1) + 1$, $\beta_i(r) = \beta_i(r-1) \cdot 0.95$ and $streak_i(r) = streak_i(r-1) + 1$ and, if i is rational, then $p_{Ci}(r) = p_{Ci}(r-1) + \alpha_w(a_i - (WB_{\mathcal{Y}} - WC_{\mathcal{T}}))$.

- With probability $(1-p_{\mathcal{A}}(r-1))P_F(r)$, the master does not audit when the set of cheaters is F. Then, according to Algorithms 9 and 10, the following updates are carried out.

  For the master: $p_{\mathcal{A}}(r) = p_{\mathcal{A}}(r-1)$ and $aud(r) = aud(r-1)$.

  For each worker $i \in W$: $v_i(r) = v_i(r-1)$.

  For each rational worker $i \in F$,

  (3) if $\rho_F(r) > \rho_{W \setminus F}(r)$ then $p_{Ci}(r) = p_{Ci}(r-1) + \alpha_w(WB_{\mathcal{Y}} - a_i)$,

  (4) if $\rho_F(r) < \rho_{W \setminus F}(r)$ then $p_{Ci}(r) = p_{Ci}(r-1) - \alpha_w \cdot a_i$,

  For each rational worker $i \notin F$,

  (5) if $\rho_F(r) > \rho_{W \setminus F}(r)$ then $p_{Ci}(r) = p_{Ci}(r-1) + \alpha_w(a_i + WC_{\mathcal{T}})$,

  (6) if $\rho_F(r) < \rho_{W \setminus F}(r)$ then $p_{Ci}(r) = p_{Ci}(r-1) + \alpha_w(a_i - (WB_{\mathcal{Y}} - WC_{\mathcal{T}}))$.

Recall that in case of a tie in the weighted majority, the master flips a coin to choose one of the answers, and assigns payoffs accordingly. If that is the case, transitions (3)–(6) apply according to that outcome.

Conditions for Eventual Correctness

We show now the conditions under which the system can guarantee eventual correctness. The analysis is carried out for a universal class of reputation functions characterized by two properties. Property 1 states that if the master audits in consecutive rounds, the aggregated reputation of the honest workers will be larger than that of cheater workers in a bounded number of rounds. Property 2 states that if the aggregated reputation of a set $X \subset W$ is larger than that of a set $Y \subset W$, then it remains so if the master audits and all workers are honest. The two properties are formally stated below.

> **Property 1:** For any constant $\delta > 0$, there is a bounded value $\gamma(\delta)$ such that, for any non-empty $X \subseteq W$ and any initial state s_r in which $v_i(r) = 0, \forall i \notin X$, if the Markov chain evolves in such a way that $\forall k = 1, \ldots, \gamma(\delta)$, it holds that $aud(r+k) = aud(r) + k$, $\forall i \in X, v_i(r+k) = v_i(r) + k$ and $\forall j \in W \setminus X, v_j(r+k) = v_j(r)$, then $\rho_X(r+\gamma(\delta)) > \delta \cdot \rho_{W \setminus X}(r+\gamma(\delta))$.

> **Property 2:** For any $X \subset W$ and $Y \subset W$, if $aud(r+1) = aud(r) + 1$ and $\forall j \in X \cup Y$ it is $v_j(r+1) = v_j(r) + 1$ then $\rho_X(r) > \rho_Y(r) \Rightarrow \rho_X(r+1) > \rho_Y(r+1)$.

Reputations Linear, Exponential and Boinc (cf. Subsection 4.4.2) satisfy Property 1, while reputation Legacy Boinc as defined does not. However, if a constant upper bound in the value of $\beta_i(r)$ is established, we obtain an adapted version of Legacy Boinc reputation that satisfies Property 1. Regarding Property 2, while reputation Exponential satisfies it, reputation Linear, Legacy Boinc and Boinc do not. The proofs of these facts are presented below. Moreover, as we show below (Theorem 4.5), this *makes a difference* with respect to guaranteeing eventual correctness.

Lemma 4.12. *Property 1 holds for reputation Linear, while Property 2 does not.*

Proof: First we show that Property 1 holds. Consider any $d > 0$, any $X \subseteq W$ non empty. Without loss of generality assume $|X| = k$. Consider rounds $r+1, \ldots r+j$, for some j, such that the master audits, workers in X are honest and workers not in X cheat. For $\forall i \in X$, $\rho_i(r+j) = \frac{v_i(r)+j+1}{aud(r)+j+2}$; and $\forall i \notin X$, $\rho_i(r+j) = \frac{v_i(r)+1}{aud(r)+j+2}$. Then $\rho_X(r+j) = \sum_{i \in X} \rho_i(r+j) = \frac{\sum_{i \in X} v_i(r)}{aud(r)+j+2} + \frac{k(j+1)}{aud(r)+j+2} \geq \frac{j+1}{aud(r)+j+2}$; and $p_{W \setminus X}(r+j) = \sum_{i \in W \setminus X} \rho_i(r+j) = \frac{(n+k)}{aud(r)+j+2} \leq \frac{n+1}{aud(r)+j+2}$. For any $j \leq \delta(n-1)$ we have, $\rho_X(r+j) \geq \frac{j+1}{aud(r)+j+2} > \frac{\delta(n-1)}{aud(r)+j+2} \geq \rho_{W \setminus X}(r+j)$. Hence, setting $\gamma(\delta) = \delta(n-1)$ proves the first part of the claim.

We now show that Property 2 does not hold. Consider any round where $aud(r+1) = aud(r) + 1$ and $\forall j \in X \cup Y$ $v_j(r+1) = v_j(r) + 1$. Without lose of generality assume that in state s_r, $|X| = k_r$. Then we have that if,

$$\rho_X(r) > \rho_Y(r)$$

$$\sum_{i\in X} \frac{v_i(r)+1}{aud(r)+2} > \sum_{i\in Y} \frac{v_i(r)+1}{aud(r)+2}$$

$$\frac{\sum_{i\in X} v_i(r)}{aud(r)+2} + \frac{k_r}{aud(r)+2} > \frac{\sum_{i\in Y} v_i(r)}{aud(r)+2} + \frac{n-k_r}{aud(r)+2}$$

$$\sum_{i\in X} v_i(r) + k_r > \sum_{i\in Y} v_i(r) + n - k_r$$

then,

$$\rho_X(r+1) > \rho_Y(r+1)$$

$$\sum_{i\in X} \frac{v_i(r)+2}{aud(r)+3} > \sum_{i\in Y} \frac{v_i(r)+2}{aud(r)+3}$$

$$\frac{\sum_{i\in X} v_i(r)}{aud(r)+3} + \frac{2k_{r+1}}{aud(r)+3} > \frac{\sum_{i\in Y} v_i(r)}{aud(r)+3} + \frac{2(n-k_{r+1})}{aud(r)+3}$$

$$\sum_{i\in X} v_i(r) + 2k_{r+1} > \sum_{i\in Y} v_i(r) + 2(n-k_{r+1})$$

Thus if $k_r \neq k_{r+1}$ then the entailment may not hold. ■

Lemma 4.13. *Property 1 and 2 hold for reputation Exponential.*

Proof:First we show that Property 1 holds. Consider any $\delta > 0$, any $X \subseteq W$ non empty. Without loss of generality assume $|X| = k$. Consider rounds $r+1, \dots r+j$, for some j, such that the master audits, workers in X are honest and workers not in X cheat. For $\forall i \in X$, $\rho_i(r+j) = \varepsilon^{aud(r)-v_i(r)}$ and $\forall i \notin X$, $\rho_i(r+j) = \varepsilon^{aud(r)+j-v_i(r)}$.Then $\rho_X(r+j) = \sum_{i\in X} \rho_i(r+j) = \sum_{i\in X} \varepsilon^{aud(r)-v_i(r)} \geq \varepsilon^{aud(r)}$ and $\rho_{W\setminus X}(r+j) = \sum_{i\in W\setminus X} \rho_i(r+j) = \sum_{i\in W\setminus X} \varepsilon^{aud(r)+j-v_i(r)} \leq (n-1)\varepsilon^{aud(r)+j}$. For any $j < -log(\delta(n-1))$ we have, $\rho_X(r+j) \geq \varepsilon^{aud(r)} > \delta(n-1)\varepsilon^{aud(r)+j} \geq \rho_{W\setminus X}(r+j)$. Hence, setting $\gamma(\delta) < -log(\delta(n-1))$ proves the claim.

Now we show that also Property 2 holds. Consider any $X \subset W$ and $Y \subset W$. Then if $\rho_X(r) = \sum_{i\in X} \varepsilon^{aud(r)-v_i(r)}$ then $\rho_X(r+1) = \sum_{i\in X} \varepsilon^{aud(r)+1-v_i(r)-1} = \sum_{i\in X} \varepsilon^{aud(r)-v_i(r)}$. If $\rho_Y(r) = \sum_{i\in Y} \varepsilon^{aud(r)-v_i(r)}$ then $\rho_Y(r+1) = \sum_{i\in Y} \varepsilon^{aud(r)+1-v_i(r)-1} = \sum_{i\in Y} \varepsilon^{aud(r)-v_i(r)}$. Thus trivially the condition $\rho_X(r) > \rho_Y(r) \Rightarrow \rho_X(r+1) > \rho_Y(r+1)$ holds. ■

Lemma 4.14. *Property 1 does not hold for reputation Legacy Boinc, unless an upper bound b of the value $\beta_i(r)$ is established;Property 2 does not hold for reputation Legacy Boinc.*

Proof: First we show that Property 1 does not hold. Consider any $\delta > 0$, and $X \subseteq W$ non empty. Without loss of generality assume $|X| = k$. Consider rounds $r+1 \dots r+j$ for some j, such that master audits, workers in X are honest and workers not in X cheat. For $\forall i \in X$ if $\beta_i(r+j) > 0.05$ then $\rho_i(r+j) = 0$ else $\rho_i(r+j) = 1 - \sqrt{\frac{\beta_i(r)\times j\times 0.95}{0.05}}$. For $\forall i \in W \setminus X$ then $\beta_i(r+j) > 0.05$ thus $\rho_i(r+j) = 0$. If $\gamma(\delta) > 0$ then $\forall i \in W \setminus X$ $\rho_i(r+j) = 0$.To have $\rho_X(r+\gamma(\delta)) > \delta\rho_{W\setminus X}(r+\gamma(\delta))$ we need to know the state s_r where the master starts to audit to know in how many rounds $\exists i \in X$ where $\beta_i(r) \leq 0.05$. Thus the condition of Property 1 for Legacy Boinc depends on the current state s_r where the master begins to audit. Hence Property 1

for reputation Legacy Boinc does not hold.

Now, we show that if an upper bound b of the value $\beta_i(r)$ is established then Property 1 holds for reputation Legacy Boinc. Consider any $\delta > 0$, and $X \subseteq W$ non empty. Without loss of generality assume $|X| = k$. Consider rounds $r+1 \dots r+j$ for some j, such that master audits, workers in X are honest and workers not in X cheat. For $\forall i \in X$ if $\beta_i(r+j) > 0.05$ then $\rho_i(r+j) = 0$ else $\rho_i(r+j) = 1 - \sqrt{\frac{\beta_i(r)\times j \times 0.95}{0.05}}$. For $\forall i \in W \setminus X$ then $\beta_i(r+j) > 0.05$ thus $\rho_i(r+j) = 0$. If $\gamma(\delta) > 0$ then $\forall i \in W \setminus X$ $\rho_i(r+j) = 0$. If a constant upper bound b in the value of $\beta_i(r)$ is established in the s_r state then in $j < \frac{0.05}{b \times 0.95}$ rounds $\forall i \in X$, $\rho_i(r+j) > 0$ and thus the condition $\rho_X(r+\gamma(\delta)) > \delta\rho_{W\setminus X}(r+\gamma(\delta))$ becomes true by setting $\gamma(\delta) < \frac{0.05}{b\times 0.95}$ and the claim is proved.

Finally, we show that Property 2 does not hold. Consider any $X \subset W$ and $Y \subset W$. If $\rho_X(r) > \rho_Y(r)$ then in the next state s_{r+1} the following apply. For $\forall j \in X \setminus Y$, $v_j(r+1) = v_j(r) + 1$ (they are honest). Thus if $\beta_j(r+1) > 0.05$ then $\rho_j(r+1) = 0$ else $\rho_j(r+1) = 1 - \sqrt{\frac{\beta_j(r)\times 0.95}{0.05}}$. Consider the following case where $\forall j \in Y$, $\rho_j(r) = 0$ and $\forall j \in X'$, where X' is the set of all workers in X besides one, lets name it z, $\rho_j(r) = 0$ and $\rho_z(r) = 1 - \sqrt{\frac{\beta_z(r)\times 0.95}{0.05}}$. Then $\rho_X(r) > \rho_Y(r)$ holds. In the next round though there is a possibility that $\forall j \in Y$, $\rho_j(r+1) = 1 - \sqrt{\frac{\beta_j(r+1)\times 0.95}{0.05}}$, while only the reputation of z changes in the next state for the X set. Thus if $1 - \sqrt{\frac{\beta_z(r)\times 0.95}{0.05}} < |Y| \times (1 - \sqrt{\frac{\beta_j(r+1)\times 0.95}{0.05}})$ then Property 2 does not hold for reputation Legacy Boinc and the claim is proved. ■

Lemma 4.15. *Property 1 holds for reputation Boinc, while Property 2 does not hold when $n > 2$.*

Proof: First we show that Property 1 holds. Consider any $\delta > 0$, and $X \subseteq W$ non empty. Without loss of generality assume $|X| = k$. Consider rounds $r+1 \dots r+j$ for some j, such that master audits, workers in X are honest and workers not in X cheat. For $\forall i \in X$ we have $\rho_i(r+j) = 1 - \frac{1}{streak_i(r+j)}$ if $streak_i(r+j) \geq 10$. For $\forall i \notin X$ $\rho_i(r+j) = 0$ since $streak_i(r+j) = 0$. Thus, $\rho_X(r+j) = \sum_{i\in X} \rho_i(r+j) = \sum_{i\in X}(1 - \frac{1}{streak_i(r+j)}) = k(1 - \frac{1}{streak_i(r+j)}) > \rho_{W\setminus X}(r+j) = \sum_{i\in W\setminus X} \rho_i(r+j) = 0$. Thus, in order for $\rho_X(r+\gamma(\delta)) > \delta\rho_{W\setminus X(r+\gamma(\delta))}$ to be true we need to set $\gamma(\delta) \geq 10$.

We now show that Property 2 does not hold when $n > 2$. Consider any round where $aud(r+1) = aud(r) + 1$ and $\forall j \in X \cup Y$ $v_j(r+1) = v_j(r) + 1$. Then, for any $X \subset W$ and $Y \subset W$ it must hold that if $\rho_X(r) > \rho_Y(r)$ then $\rho_X(r+1) > \rho_Y(r+1)$. Thus, consider state s_r where $|X| = 1$, $|Y| = n-1$ and $\forall i \in X$ $streak_i(r) \geq 10$ and $\forall i \in Y$ $streak_i(r) = 9$. It is true that $\rho_X(r) = 1 - \frac{1}{streak_i(r)} = 1 - \frac{1}{10+\delta} > \rho_Y(r) = (n-1)\cdot 0 = 0$ where $\delta > 1$. Now, in the next state s_{r+1} we have $\rho_X(r+1) = 1 - \frac{1}{streak_i(r+1)} = 1 - \frac{1}{10+\delta+1} < 1$ and $\rho_Y(r+1) = (n-1)(1 - \frac{1}{10})$ $\Rightarrow n-2 < \rho_Y(r+1) < n-1$. If $n > 2$ then $\rho_X(r+1) < \rho_Y(r+1)$ and the claim does not. Thus Property 2 does not hold for reputation Boinc when $n > 2$. ■

Moving on we present the conditions under which the system can guarantee eventual correctness, but before that we establish the terminology that will be used throughout. For any given state s, a set X of workers is called a *reputable set* if $\rho_X(r) > \rho_{W\setminus X}(r)$. In any given state s, let a worker i be called an *honest worker* if $p_{Ci}(s) = 0$. Let a state s be called a *truthful state* if the set of honest workers in state s is reputable. Let a *truthful set* be any set of truthful states. Let a worker be called a *covered worker* if the payoff of returning the correct answer is at least its aspiration plus the computing cost. I.e., for a covered worker i, it is $WB_{\mathcal{Y}} \geq a_i + WC_{\mathcal{T}}$. We refer to the opposite cases as *uncovered worker* ($WB_{\mathcal{Y}} < a_i + WC_{\mathcal{T}}$), *cheater worker* ($p_{Ci}(s) = 1$), *untruthful state* (the set of cheaters in that state is reputable), and *untruthful set*, respectively. 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 the master's algorithm, a non-zero probability of auditing is always guaranteed. This is a necessary condition. Otherwise, unless the altruistic workers outnumber the rest, a closed untruthful set is reachable, as we show in Lemma 4.16.

Lemma 4.16. *Consider any set of workers $Z \subseteq W$ such that $WB_{\mathcal{Y}} > a_i$, for every rational worker $i \in Z$. Consider the set of states*

$$S = \{s | (p_{\mathcal{A}}(s) = 0) \wedge (\forall w \in Z : p_{Cw}(s) = 1) \wedge (\rho_Z(s) > \rho_{W-Z}(s))\}.$$

Then,

(i) *S is a closed untruthful set, and*

(ii) *if $p_{\mathcal{A}}(0) = 0$, $\rho_Z(0) > \rho_{W-Z}(0)$, and for all $i \in Z$ it is $p_{Ci}(0) > 0$, then, S is reachable.*

Proof:

(i) Each state in S is untruthful, since the workers in Z are all cheaters and Z is a reputable set. Since $p_{\mathcal{A}} = 0$, the master never audits, and the reputations are never updated. From transition (3) it can be seen that, if the chain is in a state of the set S before round r, for each worker $i \in Z$, it holds $p_{Ci}(r) \geq p_{Ci}(r-1) = 1$. Hence, once the chain has reached a state in the set S, it will move only to states in the set S. Thus, S is a closed untruthful set.

(*ii*) We show now that S is reachable from the initial state under the above conditions. Because $p_{\mathcal{A}}$ and the reputations only change when the master audits, we have that $p_{\mathcal{A}}(0) = 0 \Longrightarrow p_{\mathcal{A}}(s) = 0$ and $\rho_Z(0) > \rho_{W-Z}(0) \Longrightarrow \rho_Z(s) > \rho_{W-Z}(s)$, for any state s. Malicious workers always have $p_C = 1$, and no altruistic worker may be contained in Z because $p_{Ci}(0) > 0$ for all $i \in Z$. Thus, to complete the proof it is enough to show that eventually it is $p_C = 1$ for all the workers in L, which is the set of rational workers in Z. Given that for each rational worker $j \in L$, $p_{Cj}(0) > 0$ and $WB_{\mathcal{Y}} > a_j$, from transition (3) it can be seen that there is a non-zero probability of moving

from s_0 to a state s_1 where the same conditions apply and $p_{Cj}(1) > p_{Cj}(0)$ for each rational worker $j \in L$. Hence, applying the argument inductively, there is a non-zero probability of reaching S. ■

Eventual correctness follows if we can show that the Markov chain always ends in a closed truthful set, with $p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$. We prove first that having at least one worker that is altruistic or covered rational is necessary for a closed truthful set to exist. Then we prove that it is also sufficient if all rational workers are covered.

Lemma 4.17. *If all workers are malicious or uncovered rationals, no truthful set S is closed, if the reputation type satisfies Property 2.*

Proof: Let us consider some state s of a truthful set S. Let Z be the set of honest workers in s. Since s is truthful , then Z is reputable. Since there are no altruistic workers, the workers in Z must be uncovered rational. Let us assume that being in state s the master audits in round r. From Property 2, since all nodes in Z are honest in r, Z is reputable after r. From transition (2), after round r, each worker $i \in Z$ has $p_{Ci}(r) > 0$. Hence, the new state is not truthful , and S is not closed. ■

Lemma 4.18. *Consider a reputation type that satisfies Properties 1 and 2. If all rational workers are covered and at least one worker is altruistic or rational, a closed truthful set S is reachable from any initial state. Moreover, in every state $s \in S$, $p_{\mathcal{A}}(s) = p_{\mathcal{A}}^{min}$.*

Proof: Let X be the set of altruistic and rational workers, and consider any initial state s_r. Let us define a constant $\delta = \max\{1, (1-\tau+\eta/\alpha_m)/(\tau+\eta/\alpha_m)\}$, for a fixed constant $\eta \in (0, \tau\alpha_m)$. We consider the following cases.

1. In state s_r not all the workers in X are truthful. Let us assume then that in the next $\lceil \frac{1}{\alpha_w(a_j - WP_{\mathcal{C}})} \rceil$ rounds the master audits and any worker i that has $p_{Ci} > 0$ in the round cheats. Then, from transition (1) and the fact that all rational workers are covered, after these $\lceil \frac{1}{\alpha_w(a_j - WP_{\mathcal{C}})} \rceil$ rounds all the workers in X are truthful. Then, we end up in one of the following three cases.

2. In state s_r all the workers in X are truthful, and $\rho_X(r) \leq \delta \cdot \rho_{W \setminus X}(r)$. Consider the value $\gamma(\delta)$ given in Property 1. Assume that in each of the following $\gamma(\delta)$ rounds the master audits. The workers in $W \setminus X$ are malicious, hence, it holds that $\forall \notin X : v_i(r) = 0$. Then, in these rounds all workers in X are honest (every worker in X remains truthful from transition (2) and the fact that all rational workers are covered) and all workers in $W \setminus X$ cheat because they are malicious. Therefore, it holds that $\forall i \notin X : \forall j \in [r, r + \gamma(\delta)] : v_i(j) = 0$. Then, from Property 1, after the $\gamma(\delta)$ rounds we have that $\rho_X(r + \gamma(\delta)) > \delta \cdot \rho_{W \setminus X}(r + \gamma(\delta))$. Then, we are in one of the following two cases.

3. In state s_r all the workers in X are truthful, $\rho_X(r) > \delta \cdot \rho_{W\setminus X}(r)$, and $p_{\mathcal{A}}(r) > p_{\mathcal{A}}^{min}$. Let us assume that in the next $\lceil p_{\mathcal{A}}(r)/\eta \rceil$ rounds the master audits. Then, as in the previous case, in these rounds all workers in X are honest and all workers in $W \setminus X$ cheat. Hence, the property that $\rho_X(r+k) > \delta \cdot \rho_{W\setminus X}(r+k)$ holds for each round $r+k$, for $k = 1, \ldots, \lceil p_{\mathcal{A}}(r)/\eta \rceil$. Then, by the definition of δ and the update of $p_{\mathcal{A}}$, in each round $p_{\mathcal{A}}$ is decremented by η (more precisely, by $\min\{\eta, p_{\mathcal{A}}\}$). Hence, by round $r + \lceil p_{\mathcal{A}}(r)/\eta \rceil$ it holds that $p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$. Then, we are in the following case.

4. In state s_r all the workers in X are truthful, $\rho_X(r) > \delta \cdot \rho_{W\setminus X}(r)$, and $p_{\mathcal{A}}(r) = p_{\mathcal{A}}^{min}$. Then, all subsequent states satisfy all these properties, and define the set S, independently of whether the master audits or not (from transition (2) and (6), the fact that $\delta \geq 1$, Property 2, and the update of $p_{\mathcal{A}}$). This complete the proof.

■

Now, combining Lemmas 4.17 and 4.18 we obtain the following theorem.

Theorem 4.5. *In a system where (1) the reputation type used satisfies Properties 1 and 2, and (2) all rational workers are covered, having at least one altruist or rational worker is a necessary and sufficient condition to guarantee eventual correctness. That is, from any initial state, to eventually reach a closed truthful set S where the master audits with probability $p_{\mathcal{A}}^{min}$.*

If there is no knowledge on the distribution of the workers among the three types (altruistic, malicious and rationals), the only strategy to make sure eventual correctness is achieved, if possible, is to cover all workers. Of course, if *all* workers are malicious there is no possibility of reaching eventual correctness.

4.4.5 Simulations

Our analysis has shown that reaching eventual correctness is feasible under certain conditions. Once the system enters a state of eventual correctness we are in an optimal state where the master always receives the correct task reply by auditing with a minimum probability. What is left to be clarified is under which cost eventual correctness is reached. Cost can be measured in terms of 1) reliability, 2) auditing, 3) payment to the workers and 4) time until eventual correctness is reached. Under these parameters we provide a comparison of the system's performance under the different reputation types and we are able to identify the scenarios under which every reputation type is performing best.

We present simulations for a variety of parameter combinations similar to the values observed in real systems (extracted from [6, 50]). We have designed our own simulation setup by implementing our mechanism (the master's and the workers' algorithms, including the four types of reputation discussed above) using C++. The simulations were contacted on a dual-core AMD Opteron 2.5GHz processor, with 2GB RAM running CentOS version 5.3.

General Setting

We consider a total of 9 workers as an appropriate degree of redundancy to depict the changes that different ratios of rational, altruistic or malicious workers will induce in the system. SETI-like systems usually use three workers, but using such a degree of redundancy would not allow us to present a rich account of the system's evolution. Additionally, by selecting 9 redundant workers we are able to capture systems that are more critical and aim at a higher degree of redundancy. The chosen parameters are indicated in the figures. As for the intrinsic parameter of the aspiration level we consider for simplicity of presentation 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. We set the learning rate to a small constant value, as it is discussed in [128] (called step-size there), this is the general conversion when a learning process is assumed. Thus we consider the same learning rate for the master and the workers, i.e., $\alpha = \alpha_m = \alpha_w = 0.1$. We set $\tau = 0.5$ (which means that the master will not tolerate a majority of cheaters), $p_{\mathcal{A}}^{min} = 0.01$ and $\varepsilon = 0.5$ in reputation Exponential. We use $WB_{\mathcal{Y}} = 1$, as the normalization parameter for all the results presented. Finally, the presented results are an average of 10 executions of the implementation, unless otherwise stated (when we show the behavior of typical, individual realizations). Usually we choose to depict the evolution of $p_{\mathcal{A}}$ since it is an important measure of cost for the master. In all of the depicted results in all the Figures presented here we have verified that if $p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$ then the system has already reached a state where the master receives always the correct reply, and hence eventual correctness is reached. By convention for clarity of presentation, we will simply say that the system has reached convergence once $p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$. Finally, we define $\sum_{i \in W} \rho_i S_i / |W|$ as the reputation ratio. This quantity will allow us to see the overall reputation of the workers in the system and is indicative of the existence of honest workers with higher overall reputation than cheaters.

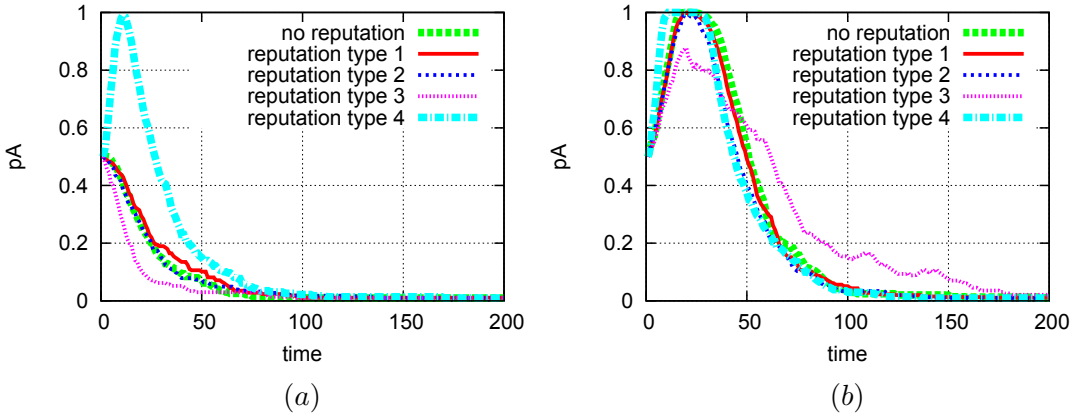


Figure 4.8: **Rational workers. Auditing probability of the master as a function of time (number of rounds)**for parameters $p_{\mathcal{A}} = 0.5$, $\alpha = 0.1$, $a_i = 0.1$, $\tau = 0.5$, $WB_{\mathcal{Y}} = 1$, $WP_{\mathcal{C}} = 0$ and $WC_{\mathcal{T}} = 0.1$. (a) initial $p_C = 0.5$ (b) initial $p_C = 1$.

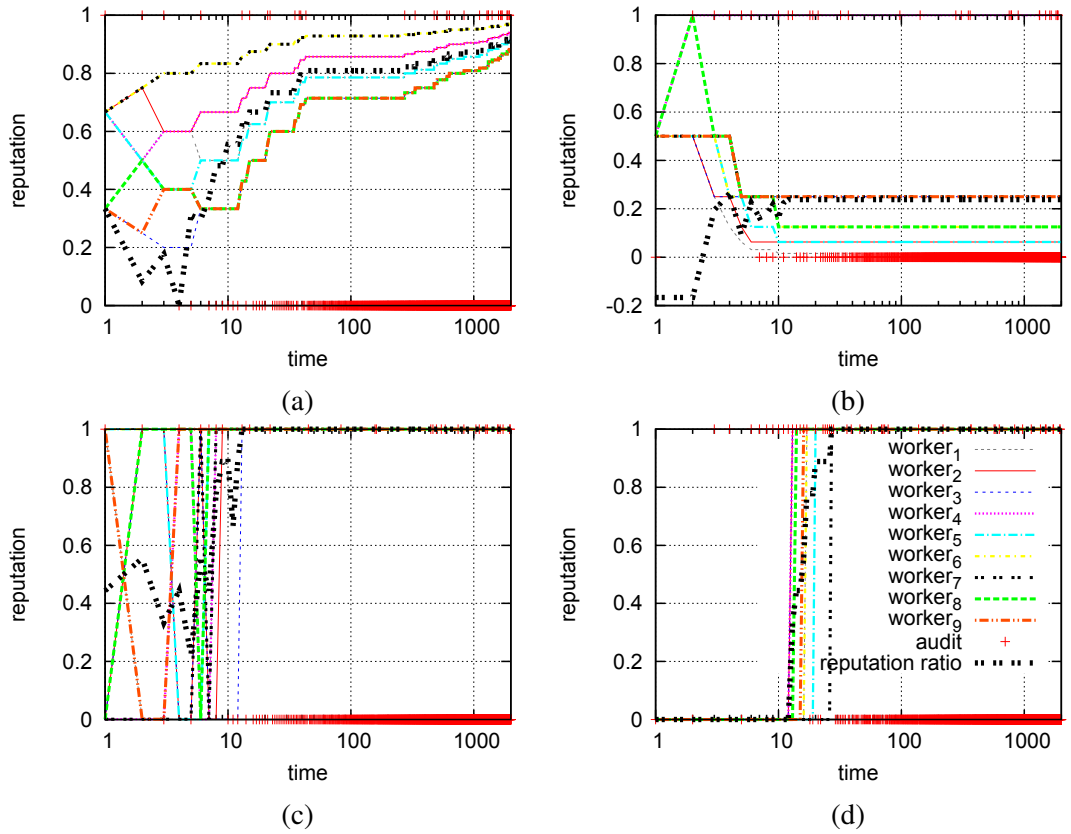


Figure 4.9: **Rational workers, for an individual realization** with initially $p_C = 0.5$, $p_{\mathcal{A}} = 0.5$, $\tau = 0.5$, $WB_{\mathcal{Y}} = 1$, $WC_{\mathcal{T}} = 0.1$, $WP_{\mathcal{C}} = 0$, $\alpha = 0.1$ and $a_i = 0.1$. (a) reputation Linear, (b) reputation Exponential, (c) reputation Legacy Boinc, (d) reputation Boinc.

We consider a variety of different scenarios where only rational workers exist in the computation or where the master decided to cover the aspired amount of payment to only a small number of workers. We also consider the case where more than one type of workers co-exist in the same computation. Finally we consider the case where some workers change type after eventual correctness is reached. Under these rich account of difference scenarios, we are able to compare the four reputation types and record the system' s behavior before eventual correctness is reached. Notice that in all the figures' legends we refer to reputation type Linear as type 1, to type Exponential as type 2, to type Legacy Boinc as type 3 and finally to type Boinc as type 4.

Presence of Only Rational Workers

In crowdsourcing systems like Amazon's Mechanical Turk the majority of workers participating in the platform are driven by monetary incentives, thus exhibiting a rational behavior where their goal is to maximize their profit. Hence, the presence of only rational workers is plausible in some real system examples. In this scenario we cover all the workers, that is, $WB_{\mathcal{Y}} > a + WC_{\mathcal{T}}$.

Figure 4.8 depicts the auditing probability of the master at each round for all 4 reputation types and the case where the mechanism designed in Section 4.3 that does not use reputation is

place. Figure 4.8 (a) shows the case where the rational workers linger between cheating or being honest in the first round of interaction by setting $p_C = 0.5$. Also the master takes an approach of ignorance by setting $p_{\mathcal{A}} = 0.5$ and not punishing the workers. Under this mild approach of the master in all 4 reputation types the system converges in roughly 100 rounds. While in the case where the master does not use reputation the system converges a bit earlier. On the other hand $p_{\mathcal{A}}$ at each round is the lowest for reputation Legacy Boinc while it is the highest for reputation Boinc. In particular for Boinc the $p_{\mathcal{A}}$ increases in the initial rounds before decreasing. This behavior is correlated with the fact that Boinc is a type that is based on a threshold, it needs 10 consecutive correct replies for a worker to increase her reputation from zero. This is also verified by the evolution of the reputation of Boinc in Figure 4.9 (d). Reputation Legacy Boinc (Figure 4.9(c)), on the other hand, allows for dramatic increases and decreases of reputation. This is a result of the indirect way we calculate reputation Legacy Boinc, as we mentioned above. Notice that in reputation Exponential (Figure 4.9(b)) reputation takes values between (0,0.3). This happens because when the master catches a worker cheating, its reputation decreases exponentially, never increasing again. Finally, reputation Linear (Figure 4.9(a)) leads rational workers to reputation values close to 1 (at a rate that depends on the value of the initial p_C) since it is a linear function.

In Figure 4.8 (b) now we assume that the workers are more aggressive towards the system starting with an initial $p_C = 1$. In this case convergence comes roughly around 120-150 rounds for reputation Linear, Exponential, Boinc and the case of no reputation. For reputation Legacy Boinc convergence comes later, roughly in 200 rounds, but $p_{\mathcal{A}}$ until convergence is lower in the first 50 rounds.

In general from Figure 4.8 we can see that the mechanism presented in Section 4.3 (without the reputation scheme) is enough to bring rational workers to produce the correct output, precisely because of their rationality. Thus if the master can be certain that only rational workers will take part in the computation is better to use the mechanism of Section 4.3. But if such a knowledge is not available then selecting reputation Exponential is the best option. Reputation Legacy Boinc performs better when $p_C = 0.5$ while it has a poor performance in the case where $p_C = 1$. As for reputation Linear it is always slightly under performing Exponential and reputation Boinc has a bad performance when $p_C = 0.5$.

Covering Only a Subset of Rational Workers

In the previous paragraph we considered only cases where the master was covering all workers, that is, $WB_{\mathcal{Y}} > a + WC_{\mathcal{T}}$ for all workers. For the case with malicious workers, as explained in Section 4.4.4, this is unavoidable if the worker's type distribution is not known. But if we know that only rational workers exist then maybe by covering only a subset of them the system can reach eventual correctness, a scenario that we now explore. Covering only a subset of the rational workers will decrease the cost of the master in terms of payment but might actually increase the cost of auditing. This precisely is the relationship we want to explore.

In Figure 4.10 the correct reply rate as a function of time is presented for the case where the

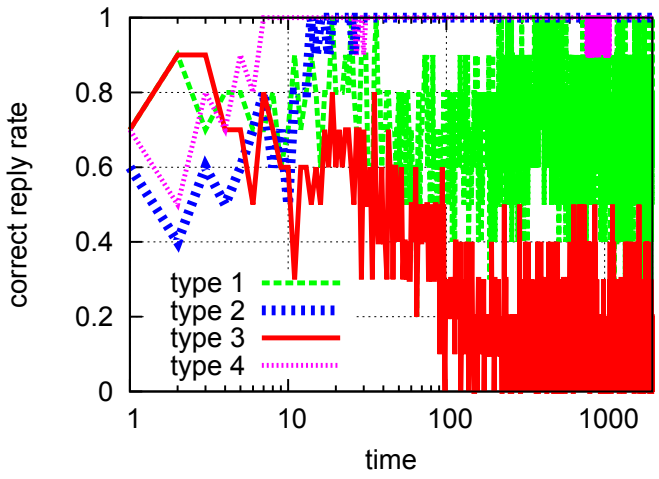


Figure 4.10: **Correct reply rate as a function of time in the presence of only rational workers.** Parameters are initial $p_{\mathcal{A}} = 0.5, WB_{\mathcal{Y}} = 1$, $WC_{\mathcal{T}} = 0.1$, $WP_{\mathcal{C}} = 0$ and $\alpha = 0.1$, $a_i = 0.1$, $\tau = 0.5$. Reputation types Linear, Legacy Boinc and Boinc have initial $p_C = 0.5$, while in Exponential, $p_C = 1$.

master covers only one worker. In a time window of 2000 rounds, as observed only reputation Exponential is able to reach eventual correctness. Even in reputation Boinc where the system looks like converged it allows the master to receive an incorrect reply. In this scenario the master has a tolerance of $\tau = 0.5$ and does not punish the workers $WP_{\mathcal{C}} = 0$. As we can see thought in other scenarios (see Figure 4.11) where the tolerance is minimum and the master punishes the workers, the situation remains the same; the system reaches eventual correctness only when Exponential is used. In the case of Linear in Figure 4.11(b1) although the master always received the correct reply, the master must always audit with $p_{\mathcal{A}}$ close to 1. The reason why only reputation Exponential is able to reach eventual correctness is because it is the only reputation type that fulfils Property 2. Under this property the reputation of the leading group of honest workers will never be able to surpass the reputation of the rest of the workers. Thus uncovered workers that periodically cheat only when the master does not audit will not be able to have a greater reputation than the covered workers. Thus the system will be able to always receive the correct reply from the covered worker that has the highest reputation than the rest of the workers' aggregated reputations, while reducing the auditing probability to minimum.

If the master covers the majority of workers then the system converges in most of the cases for different reputation types. In Figure 4.12 we can observe that reputation Exponential and Boinc converge in roughly the same amount of time in all the cases we examine. On the other hand reputation Linear does not converge in the case where the tolerance is low and the master punishes (see Figure 4.12(b1)). In this case the auditing probability is close to 1 meaning that this type of reputation was not sufficient to identify the covered rationals and form a trusted majority reputation forcing the master to audit almost at every round in order to obtain the correct reply.

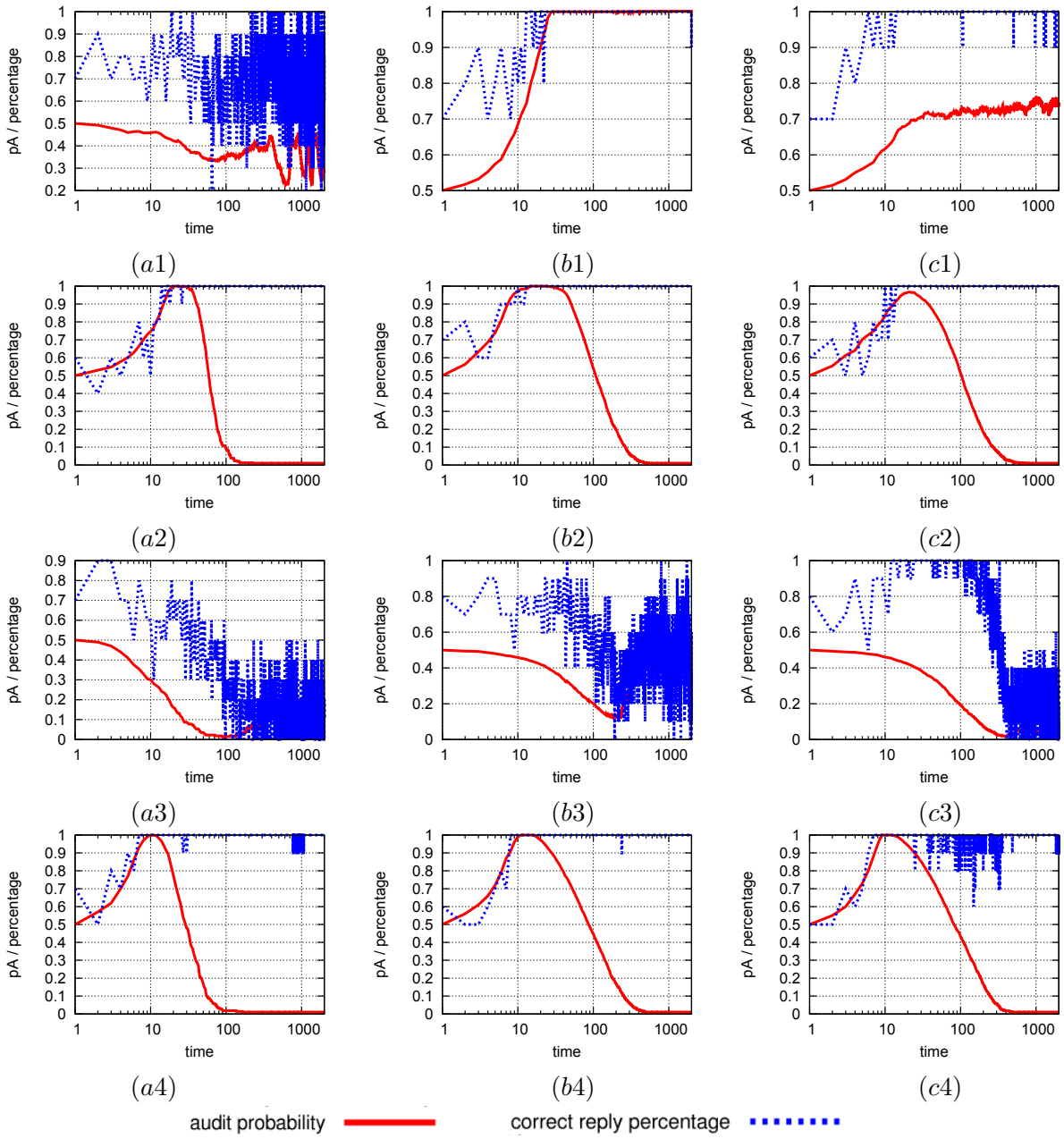


Figure 4.11: **One covered worker.** Parameters are $WC_{\mathcal{T}} = 0.1$, $a_i = 0.1$, $\alpha = 0.1$, and $WB_{\mathcal{Y}} = 0.1$ (uncovered workers) / $WB_{\mathcal{Y}} = 1$ (covered workers). Master's auditing probability, audit percentage and correct reply percentage as a function of time. First row: reputation Linear, initial $p_C = 0.5$. Second row: reputation Exponential, initial $p_C = 1$. Third row: reputation Legacy Boinc, initial $p_C = 0.5$. Fourth row: reputation Boinc, initial $p_C = 0.5$. First column $\tau = 0.5$, $WP_{\mathcal{C}} = 0$. Second column, $\tau = 0.1$, $WP_{\mathcal{C}} = 0$. Third column, $\tau = 0.1$, $WP_{\mathcal{C}} = 1$.

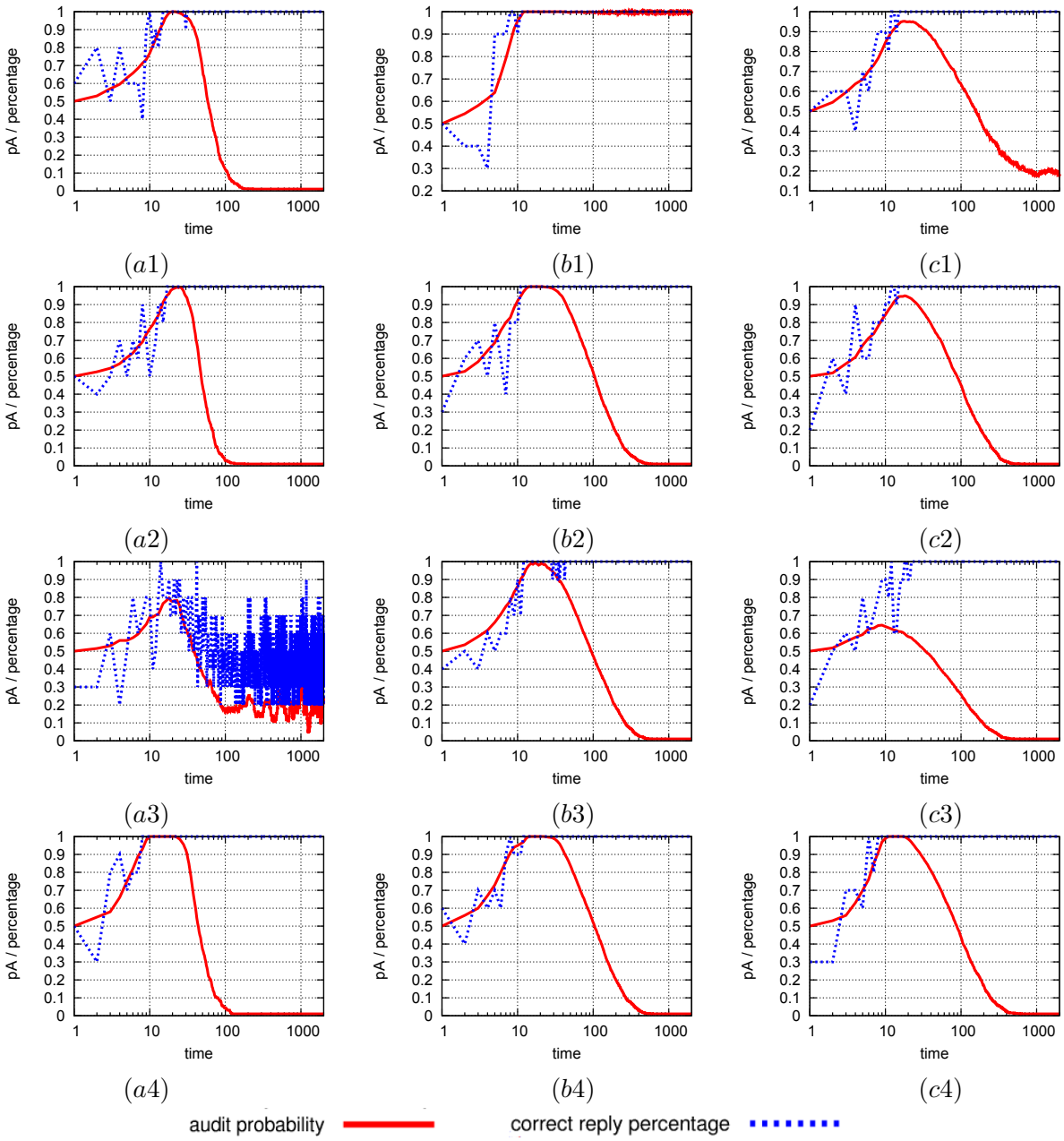
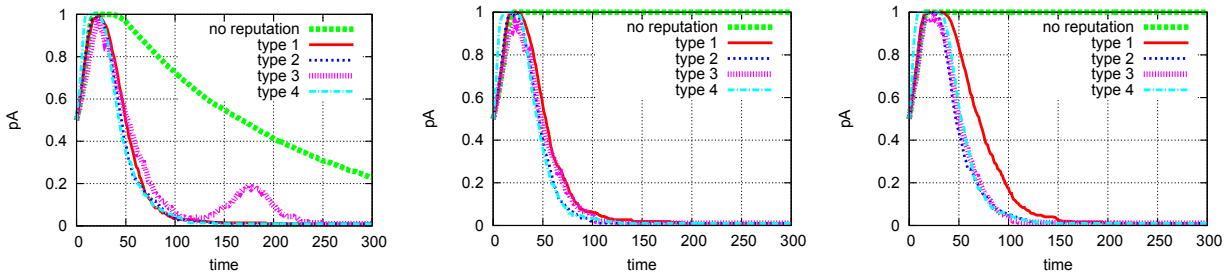


Figure 4.12: **Five covered workers.** Parameters are $WC_{\mathcal{T}} = 0.1$, $a_i = 0.1$, $\alpha = 0.1$, initial $p_C = 1$ and $WB_{\mathcal{Y}} = 0.1$ (uncovered workers) / $WB_{\mathcal{Y}} = 1$ (covered workers). Master's auditing probability, audit percentage and correct reply percentage as a function of time. First row: reputation Linear. Second row: reputation Exponential. Third row: reputation Legacy Boinc. Forth row: reputation Boinc. Left column: $\tau = 0.5$, $WP_{\mathcal{C}} = 0$. Middle column: $\tau = 0.1$, $WP_{\mathcal{C}} = 0$. Right column: $\tau = 0.1$, $WP_{\mathcal{C}} = 1$.

Also reputation Legacy Boinc was not able to converge although the auditing probability is less than a half the master does not always receive the correct task result.

This lead us to conclude that is best for the master to use reputation Exponential in the case that the master can not cover the aspiration of more than one worker. If the master can cover more than the majority of workers then both reputation Exponential and Boinc are suitable. Comparing these results (see Figure 4.11(a2) and Figure 4.12(a2)&(a4)) with the ones of Figure 4.8(b) we notice that the master does not need more auditing in the case of covering only a number of workers,what is perhaps counter-intuitive, thus our assumption was wrong. If the master is in a system where only rational workers exist then by using reputation Exponential she could guarantee eventual correctness by covering only one worker. Our analysis has indicated that a few bad cases exist where the system might actually not converge if not all the workers are covered even for Exponential. Although these are extreme cases as our simulations show, when a critical application exist that needs correctness of results such a risk can not be taken to reduce the costs of the master.



(a) (b) (c)

Figure 4.13: **Master's auditing probability as a function of time in the presence of rational and malicious workers.** Parameters in all plots, rationals' initial $p_C = 1$, master's initial $p_{\mathcal{A}} = 0.5, WB_{\mathcal{Y}} = 1$, $WC_{\mathcal{T}} = 0.1$, $WP_{\mathcal{C}} = 0$ and $\alpha = 0.1$, $a_i = 0.1$. In (a) 4 malicious and 5 rationals, (b) 5 malicious and 4 rationals , (c) 8 malicious and 1 rational.

Different Types of Workers

Moving on, we evaluate our different reputation schemes in scenarios where malicious workers exist (this was the reason for introducing reputation at the first place). Figure 4.13 shows results for the extreme case, with malicious workers, no altruistic workers, and rational workers that initially cheat with probability $p_C = 1$. We observe that if the master does not use reputation and a majority of malicious workers exist, then the master is enforced by the mechanism to audit in every round. Even with a majority of rational workers, it takes a long time for the master to reach $p_{\mathcal{A}}^{min}$, if reputation is not used. Introducing reputation can indeed cope with the challenge of having a majority of malicious workers, except (obviously) when all workers are malicious. For Linear, the larger the number of malicious workers, the slower the master reaches $p_{\mathcal{A}}^{min}$. On the contrary, the time to convergence to the $p_{\mathcal{A}}^{min}$ is independent of the number of malicious workers

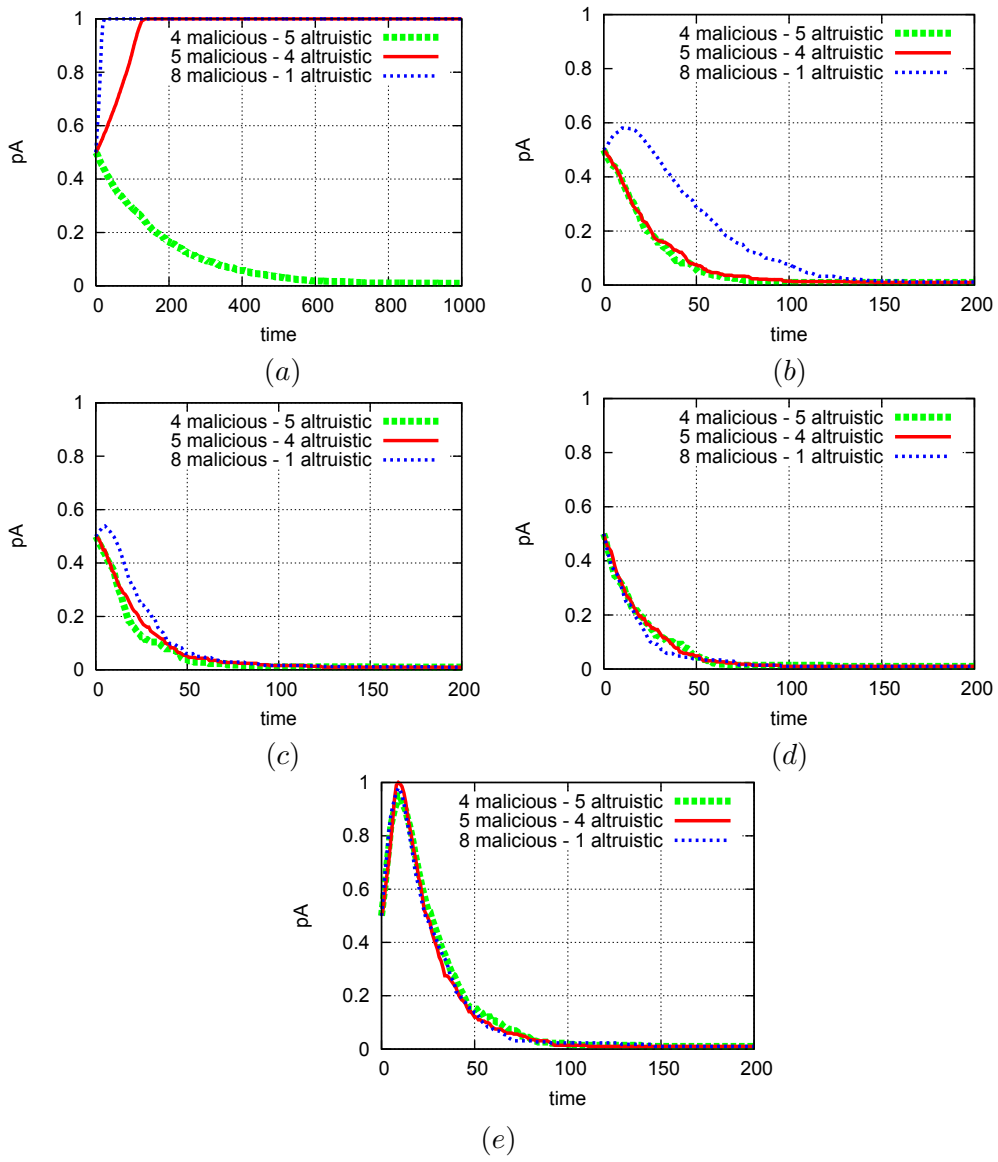


Figure 4.14: **Master's auditing probability as a function of time in the presence of altruistic and malicious workers.**Parameters in all plots, master's initial $p_{\mathcal{A}} = 0.5$, $WC_{\mathcal{T}} = 0.1$, $WP_{\mathcal{C}} = 0$ and $\alpha = 0.1$, $a_i = 0.1$. In (a) master does not use reputation, (b) master uses reputation Linear, (c) master uses reputation Exponential, (d) master uses reputation Legacy Boinc, (e) master uses reputation Boinc.

for reputation Exponential. This is due to the different dynamical behavior of the two reputations as discussed before. For reputation Legacy Boinc, if a majority of rationals exists then convergence is slower. This is counter-intuitive, but it is linked to the way reputation and error rate are calculated. On the other hand, with Legacy Boinc, $p_{\mathcal{A}}$ is slightly lower in the first rounds. As for reputation Boinc the convergence time and the behavior of the evolution of $p_{\mathcal{A}}$ is similar to reputation Exponential but in the initial rounds $p_{\mathcal{A}} = 1$ for a larger period of time, providing an additional cost to the master. Given the above observations we can conclude that reputation

Exponential has a slight advantage in all the scenarios considered (with and without a majority of rational workers) in therms of auditing cost to the master.

We have checked the behavior of the system in the case where only malicious and altruistic workers exist in the system (see Figure 4.14). As expected, if the majority of the workers is malicious and the mechanism does not use a reputation scheme the system can not converge. For the first three reputation types the mechanism converge fast and efficiently (without increasing the initial auditing probability) for the case of 4 malicious and 5 altruistic and for the case of 5 malicious and 4 altruistic. Now for the case of 8 malicious and 1 altruistic the optimum result is given by reputation Legacy Boinc while reputation Exponential has comparably good results with a slight increment of the auditing probability in the fist rounds and convergence time around the same interval. Reputation Boinc as we can see gives the worst results with the auditing probability increasing to 1 before being able to decrease. The simulations where all three types of workers co-exist are omitted since adding altruistic workers in a system with malicious and rational workers only aids the convergence of the system without providing us with any useful inside.

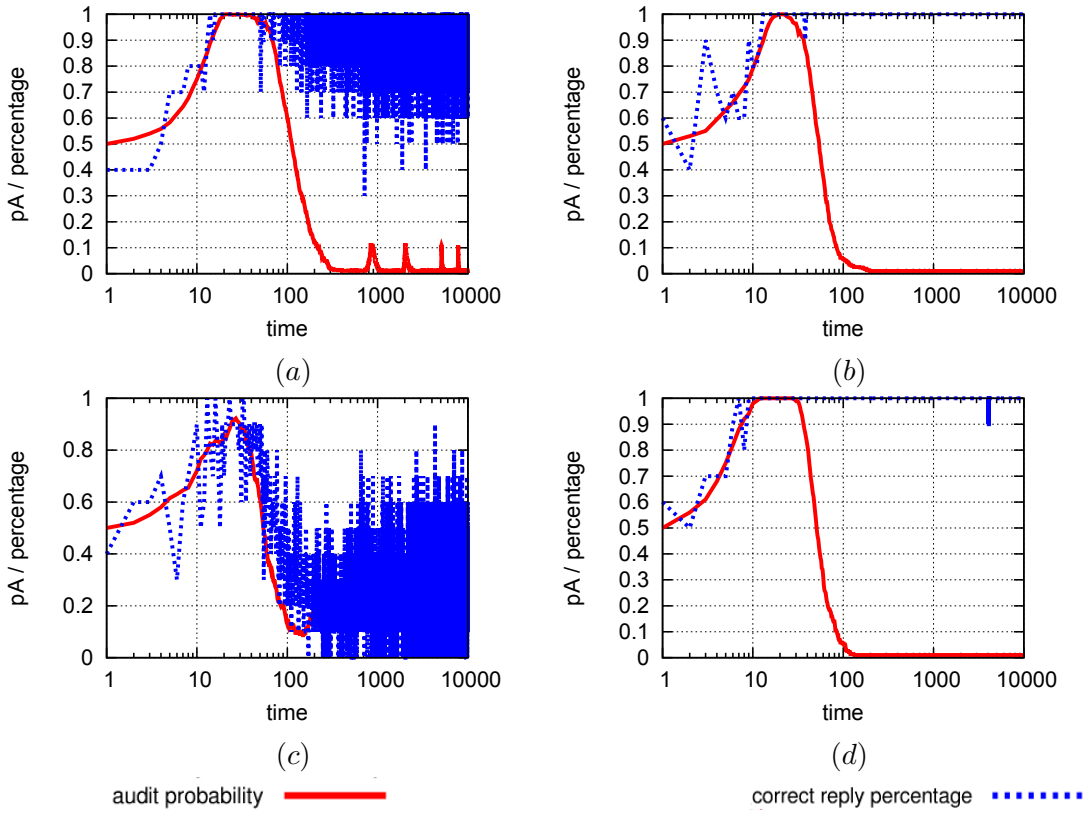


Figure 4.15: **Presence of 4 malicious and 5 rational workers, only 1 rational worker is covered.** Audit probability as function of time and correct reply percentage as a function of time. (a) Reputation Linear. (b) Reputation Exponential. (c) Reputation Legacy Boinc. (d) Reputation Boinc.

In Figure 4.15, we take a look at the case when the master decides to cover one worker out of

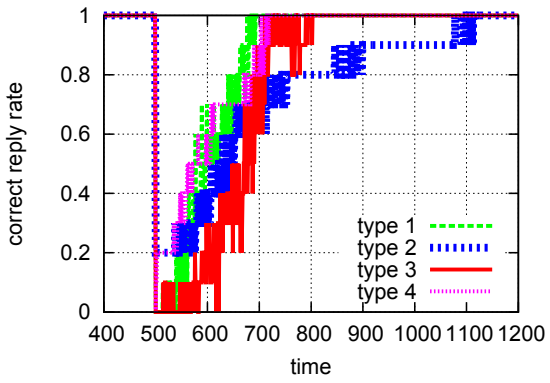


Figure 4.16: **Correct reply rate as a function of time. Presence of 5 malicious workers on the 500th round.** Parameters are initial $p_{\mathcal{A}} = 0.5, WB_{\mathcal{Y}} = 1,\ WC_{\mathcal{T}} = 0.1,\ WP_{\mathcal{C}} = 0$ and $\alpha = 0.1$, $a_i = 0.1$, $\tau = 0.5$, initial $p_C = 1$.

5 rationals and 4 malicious and that worker is rational. We notice that the system is performing in an analogous manner as in the case where all workers are covered. Only the mechanism that uses reputation Exponential is able to converge while when reputation Boinc is use the system performs quite well but still is unable to converge even after the 1000 round.

Dynamic Change of Roles

As a further check of the stability of our procedure, we now study the case when after correctness is reached some workers change their type, possibly due to a software or hardware error. We simulate a situation in which 5 out of 9 rational workers suddenly change their behavior to malicious at time 500, a worst-case scenario. Figure 4.16 shows that after the rational behavior of 5 workers turns to malicious, convergence is reached again after a few hundred rounds and eventual correctness resumes. As we see from Figure 4.17, it takes more time for reputation Exponential to deal with the changes in the workers' behavior, because this reputation can never increase, and hence the system will reach eventual correctness only when the reputation of the workers that turned malicious becomes less than the reputation of the workers that stayed rational. It also takes more time for reputation Legacy Boinc to deal with the changes in the workers' behavior (see Figure 4.17). In the case of reputation Linear, not only the reputation of the workers that turned malicious decreases, but also the reputation of the workers that stayed rational increases. As for reputation Boinc it take a bit more time than reputation Linear to reach eventual correctness again after the change of behavior.

Therefore, reputation Linear exhibits better performance in dealing with dynamic changes of behavior than reputation types Exponential, Legacy Boinc and Boinc. As Figure 4.18 depicts though the auditing probability of Linear and Legacy Boinc is drastically increasing when the change of behavior happens while for reputation Exponential and Boinc this is not the case.

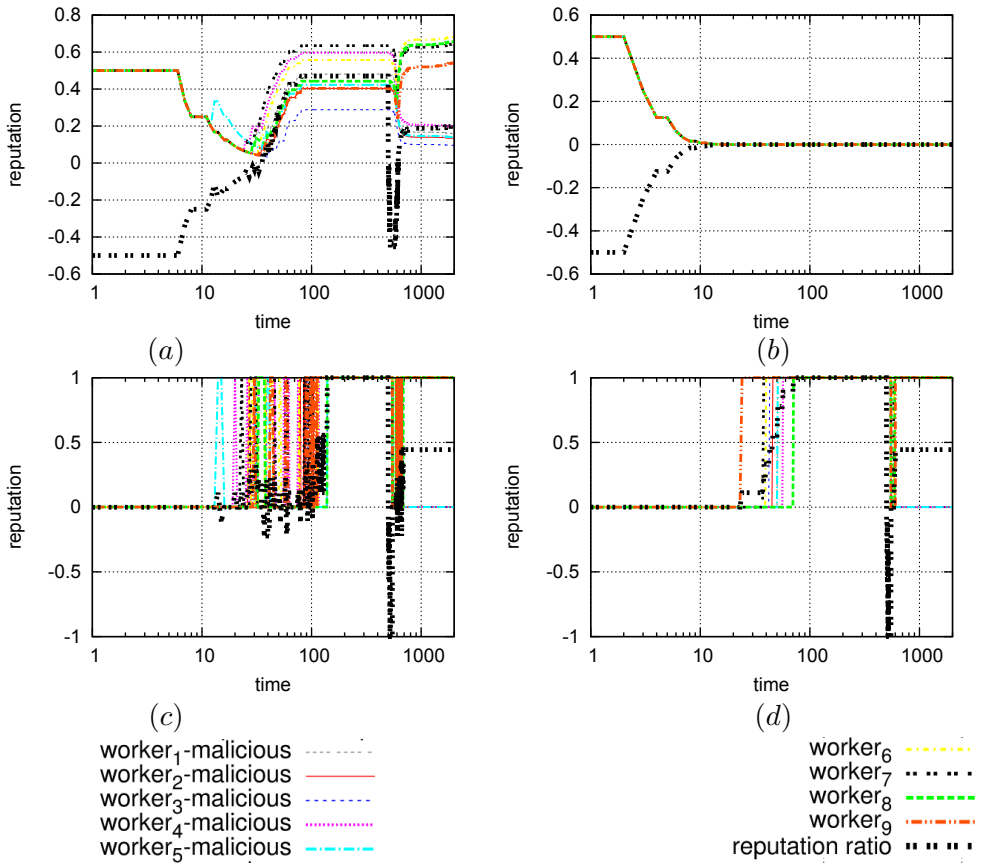


Figure 4.17: **Presence of 5 malicious workers on the 500th round.** Workers' reputation as a function of time, audit occurrences as a function of time and reputation ratio as a function of time, for an individual realization. Parameters in all panels, initial $p_C = 1$, initial $p_{\mathcal{A}} = 0.5$, $WC_{\mathcal{T}} = 0.1$, $WP_{\mathcal{C}} = 0$ and $\alpha = 0.1$, $a_i = 0.1$, $\tau = 0.5$. (a) Reputation Linear, (b) reputation Exponential, (c) reputation Legacy Boinc and (d) reputation Boinc.

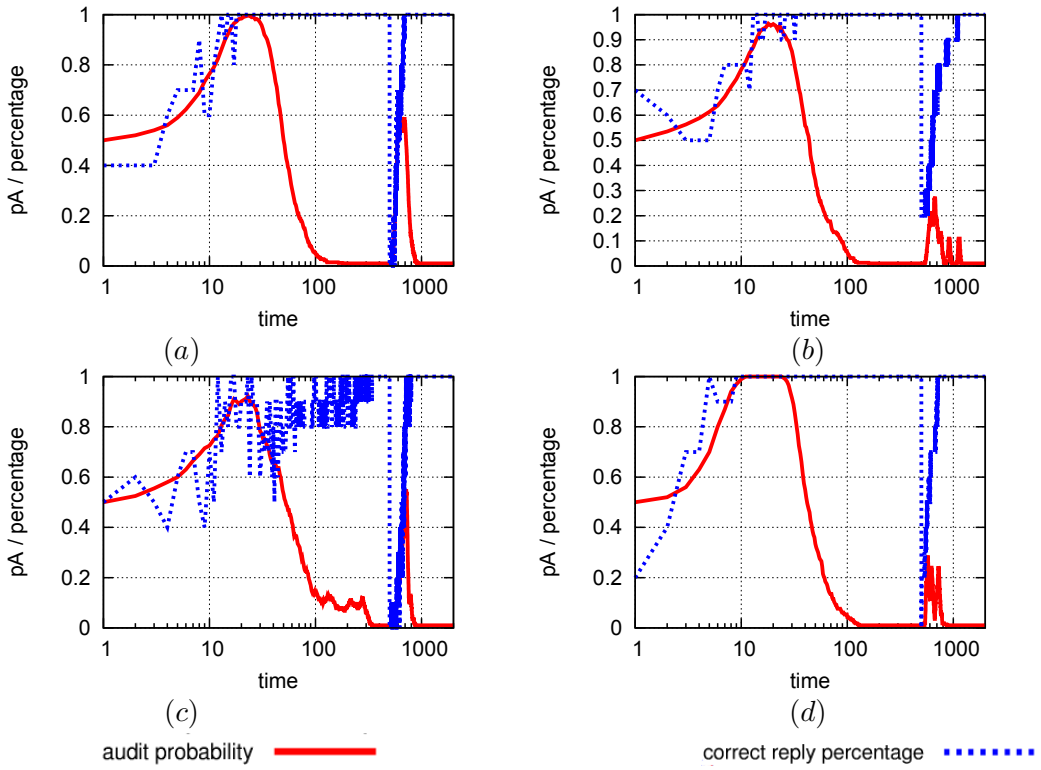


Figure 4.18: **Presence of 5 malicious workers on the 500th round.** Audit probability as a function of time and correct reply percentage as a function of time. Parameters in all panels, initial $p_C = 1$, initial $p_{\mathcal{A}} = 0.5$, $WC_{\mathcal{T}} = 0.1$, $WP_{\mathcal{C}} = 0$ and $\alpha = 0.1$, $a_i = 0.1$, $\tau = 0.5$. (a) Reputation Linear, (b) reputation Exponential and (c) reputation Legacy Boinc, (d) reputation Boinc.

4.5 Selecting from a Pool of Workers

4.5.1 Introduction

In the previous two sections we implicitly assumed that workers are randomly split into groups of size n. For each of these group of workers, the behavior of the rational workers is being reinforced by the master, while the influence of the malicious workers to the task outcome of a round is diminished, through the use of reputation schemes. This is one approach to address the problem. On one hand the complexity of the problem is reduced but on the other hand eventual correctness with minimum auditing can not be guaranteed in every group of workers. For instance, there is a positive probability that only malicious workers are selected and the master would have to audit always to obtain the correct result.

Another approach to the problem is to assume that the master can select in every round the most reliable workers from all the available workers. The advantage of this approach is that possible unresponsive workers can be avoided, since workers are selected from a huge set of workers, that we call "pool". Unlike the assumptions in the previous two sections (and most previous literature), in practice workers are not always available. For instance, Heien et al. [63] have found that in BOINC [8] 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.

We consider the existence of a pool of N workers out of which the master chooses $n < N$ and workers that regardless of their behavior (malicious, altruistic or rational) might be unavailable (e.g., be disconnected, be busy performing other tasks, etc). Building on the previous sections, we model the behavior of the rational workers through reinforcement learning. Additionally, we use a reputation scheme, not only to decide on the correct task response but also for selecting the most reputable and most responsive worker from the pool. Given the designed mechanism, 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.

We consider that the approach discussed in this subsection, could be almost directly applicable to the crowdsourcing example. The mechanism presented in this section is novel in two fronts: (*i*) it leveraged 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.

Contributions

- 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 truth-

fulness reputation. Namely, Boinc reputation (inspired in the "adaptive replication" of BOINC), Exponential reputation (that we introduce in this work), and Linear reputation (inspired on the work of Sonnek et al. [127]). All the above are included in the Subsection 4.5.2.

- We present a mechanism, in Subsection 4.5.3, 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.

- We also show formally, in Section 4.5.4, 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 4.5.5 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.

4.5.2 Model

In this subsection we build on top of the general model presented in Section 4.2 to capture the new approach presented. We give an updated definition of the master-worker framework, of rational workers' cheating probability and of the eventual correctness. Moreover, we formally define the concept of an unavailable worker and of truthfulness and responsiveness reputation.

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* $r = 1, 2, ...$, like in the general model presented. 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 assigned a task to compute, but may not do so (e.g., they are unavailable). 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. A reputation mechanism is employed by the master to choose the n most reputable workers in every round.

Worker Unavailability

In Internet-based master-worker computations, and especially in volunteering computing, workers are not always available to participate in a computation [63] (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

Like in the general model (see Section 4.2) we assume the presence of altruistic and malicious workers that 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 define the probability of a rational worker i cheating, with a slightly different notation to capture the fact that the rational worker is selected from the pool. Thus we denote by $p_{Ci}(r)$ the probability of a rational worker i cheating in round r, provided that $i \in W^r$.

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, in this section, 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 [127]) the reputation changes at a linear rate. The

Exponential reputation type (introduced in Section 4.3) 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 [11]. In the BOINC system this reputation method is used to avoid redundancy if a worker is considered honest[2].

Recall that, in the previous section we also considered an adaptation of a legacy adaptive replication method used in BOINC, refereed to as Legacy Boinc. We have seen though, that considering also this legacy type did not improve the performance of our mechanism, in terms of eventual correctness. Thus, we decided to omit Legacy Boinc in this section and turn our focus to more engaging reputation types.

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

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.

[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.

Algorithm 11 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*

4.5.3 Reputation-based Mechanism

We now present our reputation-based mechanism. Like in the previous two sections, 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 11, 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

Algorithm 12 Algorithm for Rational Worker i

```
1  p_Ci ← y, where y ∈ [0, 1]
2  repeat forever
3      wait for a task T from the master
4      if available then
5          decide whether to cheat or not independently with distribution P(cheat) = p_Ci
6          if the decision was to cheat then
7              send arbitrary solution to the master
8              get payoff Π_i
9              p_Ci ← max{0, min{1, p_Ci + α_w(Π_i − a_i)}}
10         else
11             send compute(T) to the master
12             get payoff Π_i
13             p_Ci ← max{0, min{1, p_Ci − α_w(Π_i − WC_T − a_i)}}
```

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 11, 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 $\Pi_i = -WP_{\mathcal{C}}$; if i is honest, then $\Pi_i = WB_{\mathcal{Y}}$. If the master does not audit, and i returns the value of the weighted majority (i.e., $i \in R_m$), then $\Pi_i = WB_{\mathcal{Y}}$, otherwise $\Pi_i = 0$.

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 [128]. 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 12. 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 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 [128], 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.

4.5.4 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.

Observation 4.1. *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.*

Proof: Let W^1 be the subset of n workers chosen by the master in the first round. Since initially there is no knowledge on the type of each worker, there is a positive probability p that all workers in W^1 are malicious. By assumption $W^r = W^1$ for all $r > 1$, and hence there is a probability $p > 0$ that the workers are chosen by the master in each round are all malicious. Assume this happens, then we claim that eventual correctness cannot be satisfied. Assume otherwise; hence, by definition of eventual correctness, there is a round r_0 such that in all rounds $r \geq r_0$ the master uses $p_{\mathcal{A}} = p_{\mathcal{A}}^{min} < 1$. But then, the probability that the master obtains the correct task result in round r cannot be 1, as required by the eventual correctness property, since with probability $1 - p_{\mathcal{A}} > 0$ all the received replies are incorrect and the master does audit them. Hence, eventual correctness cannot be satisfied. For the sake of contradiction, assume that the master does not change workers over rounds and eventual correctness is achieved. That is, there is a round r_0 such that for all rounds $r \geq r_0$ the master uses $p_{\mathcal{A}} = p_{\mathcal{A}}^{min} < 1$ and obtains the correct answer with probability 1, even though the master never changes workers. Let W be the subset of n workers chosen by the master that will never change. Given that the type of each worker is unknown, that workers are chosen uniformly at random, and that there are at least n malicious workers, there is a probability $p > 0$ that the master chooses only malicious workers. Consider round r_0. In round r_0, there is a probability $1 - p_{\mathcal{A}} > 0$ that the master does not audit. Thus, the probability that the master obtains the correct answer in round r_0 is $1 - p(1 - p_{\mathcal{A}}) < 1$, which is a contradiction. ■

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

Proof: For the sake of contradiction assume that every non-malicious worker i has $d_i < 1$ and eventual correctness is satisfied. Then, by definition of eventual correctness, there is a round r_0 such that in all rounds $r \geq r_0$ the master uses $p_{\mathcal{A}} = p_{\mathcal{A}}^{min} < 1$. In any round $r \geq r_0$ there is a positive probability that the master does not audit and all the replies received (if any) are incorrect. Then, there is a positive probability that the master does not obtain the correct task result, which is a contradiction. ■

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 11.

Theorem 4.6. *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 11 is used with the responsiveness reputation and any of the truthfulness reputations Linear or Exponential.*

Proof: First, observe that the responsiveness reputation of worker i will always be $\rho_{rs_i} = 1$, since $d_i = 1$. In fact, all workers j with $d_j = 1$ will have responsiveness reputation $\rho_{rs_j} = 1$ forever. Moreover, for any worker k with $d_k < 1$ that is selected by the master an infinite number of rounds, with probability 1 there is a round r_k in which k is selected but the master does not receive its reply. Hence, $\rho_{rs_k}(r) < 1$ for all $r > r_k$.

Let us now consider truthfulness reputation (of types Linear and Exponential). All altruistic workers j (including i) have truthfulness reputation $\rho_{tr_j} = 1$ forever, since the replies that the master receives from them are always correct. Malicious workers, on the other hand, fall in one of two cases. A malicious worker k may be selected a finite number of rounds. Then, there is a round r'_k after which it is never selected. If, conversely, malicious worker k is selected an infinite number of rounds, since $d_k > 0$ and $p_{\mathcal{A}} \geq p_{\mathcal{A}}^{min} > 0$, its replies are audited an infinite number of rounds, and there is a round r'_k so that $\rho_{tr_k}(r) < 1/n$ for all $r > r'_k$.

Hence, there is a round R such that, for all rounds $r > R$, (1) every malicious worker k has $\rho_{tr_k}(r) < 1/n$ or is never selected by the master, and (2) every worker k with $d_k < 1$ has $\rho_{rs_k}(r) < 1$ or is never selected by the master. Since there is at least worker i with reputation $\rho_i = 1$, we have that among the n workers in W^r, for all rounds $r > R$, there is at least one altruistic worker j with $d_j = 1$ and $\rho_j = 1$, and the aggregate reputation of all malicious workers is less than 1. Hence, the master always gets correct responses from a weighed majority of

workers. This proves the claim. ■

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 4.7. *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 11 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.*

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 4.7 shows a positive correlation among a truthfulness reputation with the availability factor of a worker in the case a large number of altruistic workers.

4.5.5 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 [6, 50], and are rather similar to the ones used in Subsection 4.4.5. We have developed our own simulation setup by implementing our mechanism (Algorithms 11 and 12, 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 [128] (called step-size there), is generally set to a small constant value for practical reasons. We set $\tau = 0.5$, $p_{\mathcal{A}}^{min} = 0.01$, 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 $p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$)[1]. 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 section 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 4.19 (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 4.19 (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 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 example 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 4.19 (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.

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

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 4.20 (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 4.20 (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 4.20 (a1)-(b1), in all cases setting $p_{\mathcal{A}} = 0.5$ initially improves the performance of the mechanism.

The results of Figure 4.20 (a1)-(b1) are confirmed by Theorem 4.6. 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 4.7 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 depicted 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 4.20 (a2)-(b2), depicts more scenarios with different workers types ratios, in the pres-

ence 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.

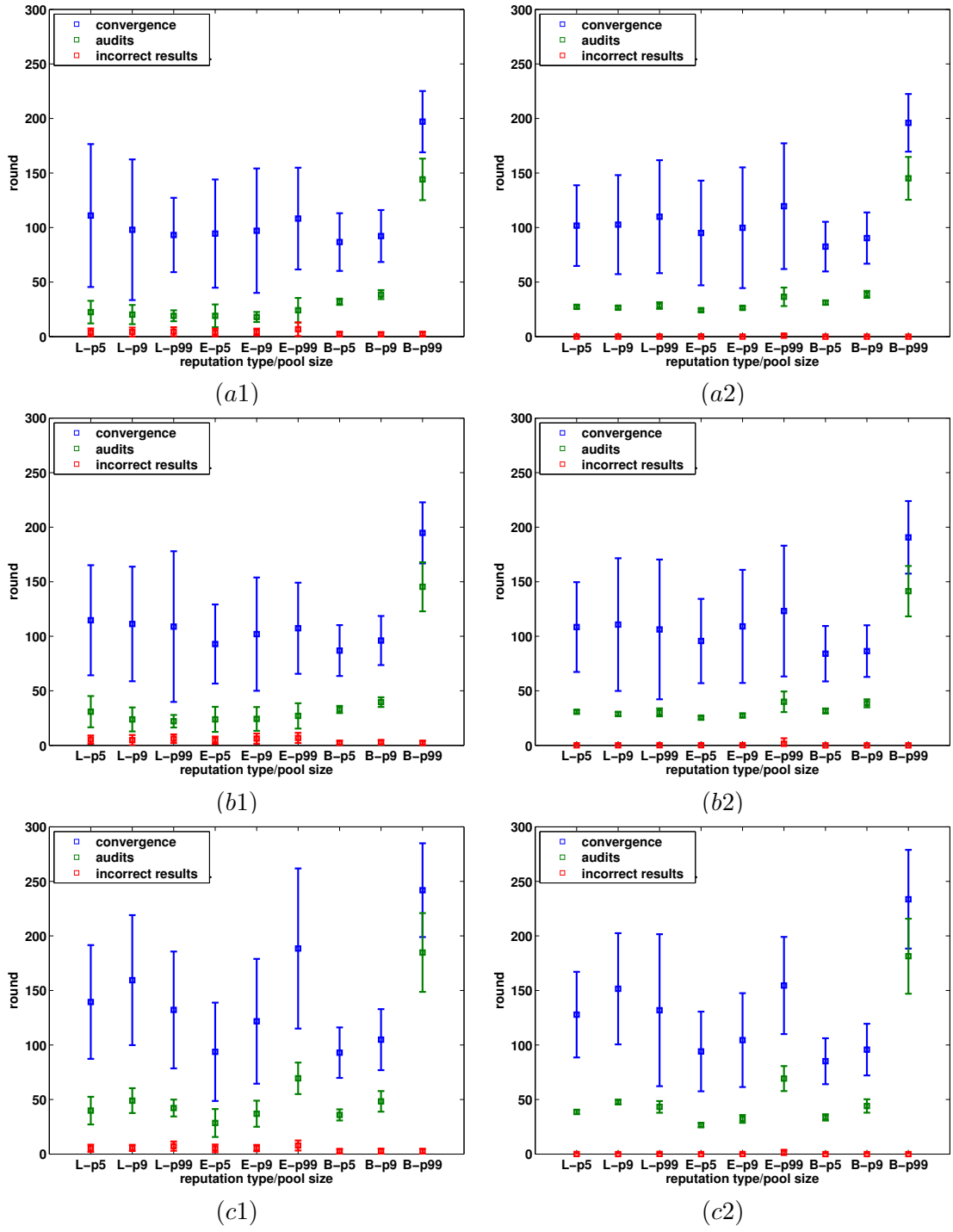


Figure 4.19: **Simulation results with full availability.** First column plots are for initial $p_{\mathcal{A}} = 0.5$. Second column plots are for initial $p_{\mathcal{A}} = 1$. The bottom (red) errorbars present the number of incorrect results accepted until convergence ($p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$), 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.

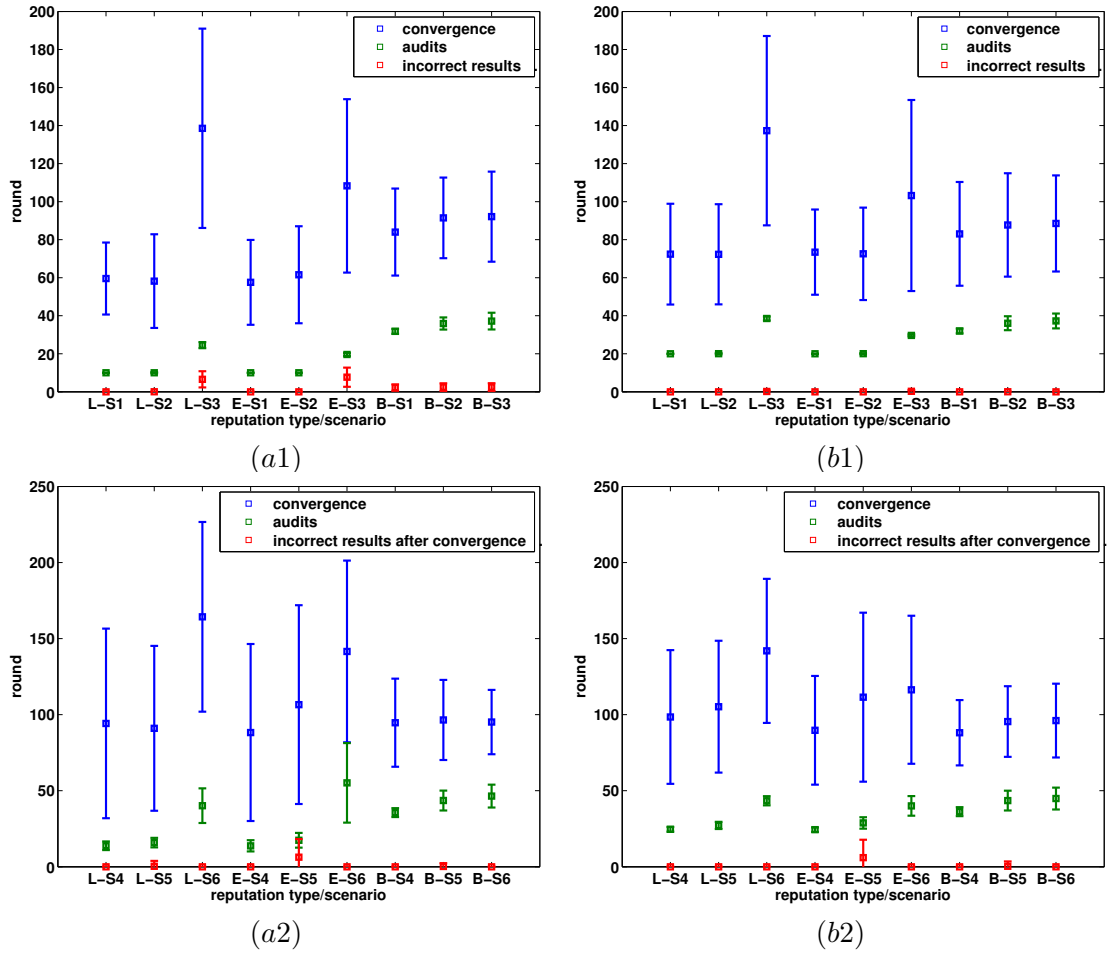


Figure 4.20: **Simulation results with partial availability: (a1)-(a2) initial** $p_{\mathcal{A}} = 0.5$**, (b1)-(b2) initial** $p_{\mathcal{A}} = 1$ **.** For (a1)-(b1) The bottom (red) errorbars present the number of incorrect results accepted until convergence ($p_{\mathcal{A}} = p_{\mathcal{A}}^{min}$). 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 $d = 0.5$.

Chapter 5

Worker Characterization: An Experimental Evaluation

5.1 Introduction

The preceding chapters addressed the issue of reliability in Internet-based task computing systems and proposed solutions that deal with the untrustworthy nature of the workers. To this respect two model assumptions are designed that characterize the workers' behavior. Recall that workers are characterized as following an altruistic or troll behavior subject to errors in the error probability model, while in the rationality model the presence of malicious, altruistic and rational workers is assumed. These assumptions are based on the literature [8,9,44,63,73,77,94,123,134] describing the errors on volunteer computing or the low quality of task results in crowdsourcing systems.

Crucially, the behavior of workers in volunteer computing is easier to be observed since the tasks are almost always in the form of a machine executable code and errors can be classified [77] but this is not the case in crowdsourcing. Crowdsourcing tasks in platforms like Amazon Mechanical Turk (AMT) are directed to humans, so mistakes can arise more easily, and it is more difficult to classify the nature of the error, or the intended behavior of the workers behind that error. In this chapter we evaluate our two modeling assumptions, regarding the nature of the workers, on micro-task crowdsourcing platforms and in particular on AMT.

There is a huge amount of literature referring to the demographics of the participants [27,105] trying to understand the characteristics of this population [104] the incentives that can improve the quality, reliability and accuracy of the results [27, 93, 107, 114, 123]. In particular, there are some very enlightening works [66, 94] on the main incentive that potentially could drive workers quality, that is payments and a work by Kaufmann et al. [74] describing the main motivations for participating in a crowdsourcing platform. Although this huge amount of literature can shed light to the motives related to the workers behavior, the purpose of our experiments is a slightly different one. We wish to study whether our characterization on the workers behavior is a valid

one, that is if workers participating in our experiments can be characterized either through the error probability mode or through the rationality model.

AMT is developed to bring together requesters having a set of tasks to perform with workers that are willing to perform a task under payment. A requester announces her task on the platform in the form of a Human Intelligence Task (HIT), as tasks are called in AMT, together with additional information on the task. Workers that like the task description accept to perform the HIT and report back a result. Requesters evaluate the task results and reward the workers. In this experimental evaluation we took the role of the requester and we have posted HITs with different degrees of difficulty, which were executed by workers on the platform. Based on the data collected, we try to identify the relationship among the time invested by a worker, the difficulty of the task and the correctness of the task result. Moreover, we aim at identifying workers with a guessing behavior. Given the evalutation of the received results we take one step forward and we try to characterize the workers in different types, confirming our two model assumptions on the workers' behavior. Interestingly, this work will have implications not only for the issue of the reliability of crowdsourcing results, but even for the reliability of economic and psychological experiments done with AMT; in fact, only in 2015 more than 1100 studies of this type were carried out with a pool of about 30000 AMT subjects [25]. As we will see below, the population may be distributed in types that may affect the results in these two contexts.

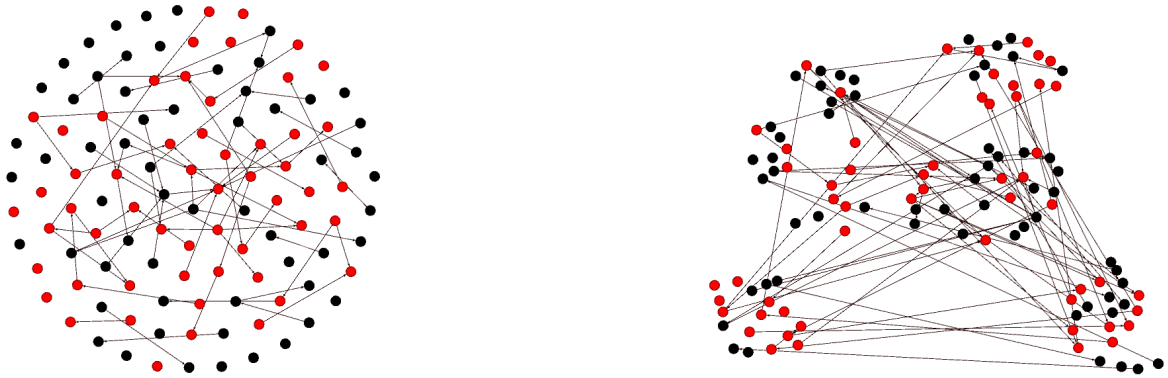
5.2 Experiments on Amazon Mechanical Turk

The main objectives of this set of experiments, are to (a) identify an intrinsic behaviors through assessing the correlation among the worker's accuracy with the response time and difficulty of the task, (b) identify different types of workers that characterize their behavior. We begin by a assessing a number of hypothesis, while we also conjecture that a set of workers reply with a guessed answer.

To test our hypotheses we have designed three variations of the same tasks. The general task designed consists of four subtasks, each of which shows a connected graph to the workers and asks them a question. The four graphs presented are the same in all three variations of the task. More precisely, the workers are presented with two simple graphs which are different perspectives (nodes are distributed in a different way) of the same graph. The other two graphs presented to the workers are more complex, and again are two different perspectives of the same graph. The graphs shown to the workers are depicted in Figure 5.1. Note that the graphs appear to each worker in a random order. The task varies depending on the question asked related to the graphs illustrated to the worker. The three task variations we consider are the following:

Color: For each of the four graphs, we ask the workers if the red nodes or the black nodes are the majority in the graph.

Majority: For each of the four graphs, we inform the workers that the majority of the received



$(a1)$ $(a2)$

$(b1)$ $(b2)$

Figure 5.1: **Graphs used in the tasks given to the workers.** (a1) subtask with graph G1, (a2) subtask with graph G2, (b1) subtask with graph G3 and (b2) subtask with graph G4. In (a1) and (a2) there are 59 black nodes and 55 red nodes; in (b1) and (b2) there are 28 black nodes and 14 red nodes.

answers claims that nodes with a particular color form a majority in the graph. We ask the worker to let us know whether she agrees with the majority and in case she does not agree we ask the worker to count the number of nodes in the majority color.

Count: For each of the four graphs, we ask the workers to let us know the total number of nodes in the graph.

The idea behind these three HITs is to evaluate the workers behavior on a task with a binary answer, that is HIT color. For the same task evaluate the workers behavior when they have the possibility to provide a precise solution or choose the default solution a.k.a, HIT majority, and

finally evaluate the workers answers on the same task but asking them to provide a precise answer in the HIT count. Hence, we like to study the behavior of the workers when we present them with an easy way to avoid investing a lot of time in the task, that is to count the nodes, and yet have a high chance of providing the correct reply. Thus, these experiments try to simulate actual tasks posted on AMT that have a binary answer, a default answer or ask for a precise answer (and thus multiple distinct replies are received). In the rest of the section we describe how we implemented these tasks on AMT and we present our findings.

5.2.1 Experimental Set-Up

As we mentioned above our designed tasks are implemented on AMT, where each task given to the workers is called a HIT. Three different HITs were implemented corresponding to the three task variations. Below we present a comprehensive report of the values assigned to the HIT parameters, such as the task description, promised reward, number of assignments, allocated time, etc. Besides these parameter values which we are set before placing the HIT online, we have a number of internal task parameters. In particular, we record the worker's response to each of the four subtasks together with the time it took the worker to respond. Moreover, we record the workers responses on a few demographic questions: age, gender, education and occupation.

Notice that the payment in each HIT varies proportionally to its difficulty, in an effort to cancel out the effect that payments might have on motivating the workers honesty (or equivalently dishonesty). The three HITs designed are presented below in order of ascending task difficulty.

Color HIT Design

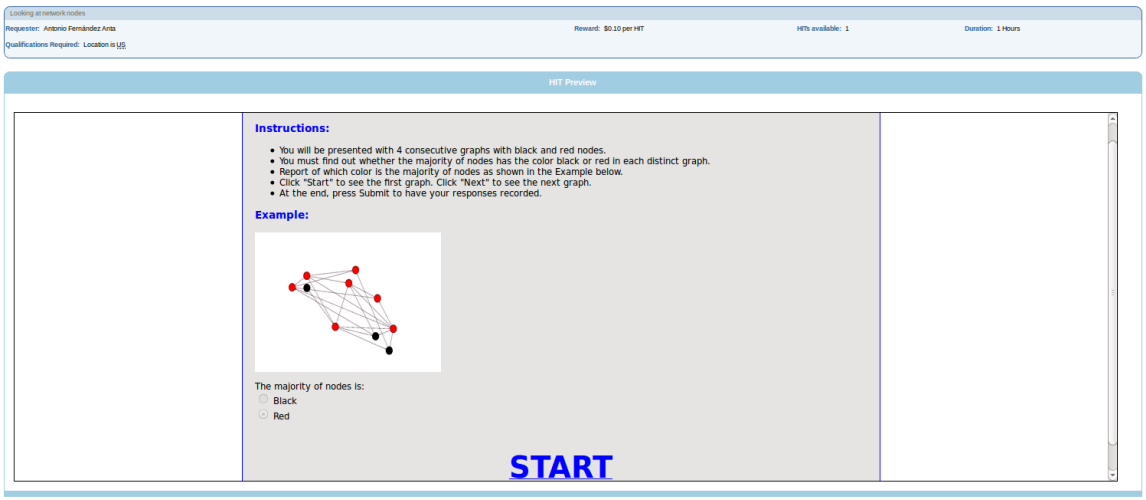


Figure 5.2: **HIT preview, for the task variation color, as seen by the worker.**

The initial preview of the HIT implementing the color variation is presented in Figure 5.2, while the HIT parameters are the following:

Title: Looking at network nodes;
Description: Find out whether the majority of nodes has the color black or red in a graph;
Keywords: graph, network, majority;
Rewards per assignment: $0.10;
Number of assignments per HIT:100;
Time allotted per assignment: 1h;
HIT expires in: 7 days (with possibility of extending the initial expiration time);
Auto-approve and pay Workers in: 7 days;
Require that Workers be Masters to do your HITs: No;
Workers must: be from USA;
Workers must: not have participated in our experiments in the past.

If the workers accept to compute the task based on the HIT preview (see Figure 5.2) that gives them the instructions and an example of the task, then the HIT is presented to them, and the four graphs of Figure 5.1 are shown to them in a random order. No default value, red or black is preselected for the workers. Notice that is not obvious which is the majority color for graph in Figure 5.1 G1 while for G2 graph it is more obvious and for graphs G3 and G4 the majority color in the nodes is obvious.

Note that we do not provide any explanation to the workers as to the methodology we will use to reward them (based on the validity of all 4 replies or proportionally). Workers are only aware that completing correctly a task, according to the requesters standards, will be worth $0.10. We use such an approach for the following reasons: (1) To simulate the standard "vague" approach taken by the majority of the requesters. (2) To avoid workers strategically replying to questions if we were to reward proportionally for each correct answer to a subtask. (3) In case we were to reward only if a worker replied correctly to all questions, we would have actually given additional incentives to the workers to respond honestly, while our intention is to study their behavior.

Majority HIT Design

The initial preview of the HIT implementing the majority variation is presented in Figure 5.3, while the HIT parameters are the following:

Title: Looking at network nodes;
Description: Let us know your opinion on the majority color of the graphs shown;
Keywords: graph, network, majority;
Rewards per assignment: $0.15;
Number of assignments per HIT:100;
Time allotted per assignment: 1h;
HIT expires in: 7 days (with possibility of extending the initial expiration time);
Auto-approve and pay Workers in: 7 days;
Require that Workers be Masters to do your HITs: No;
Workers must: be from USA;

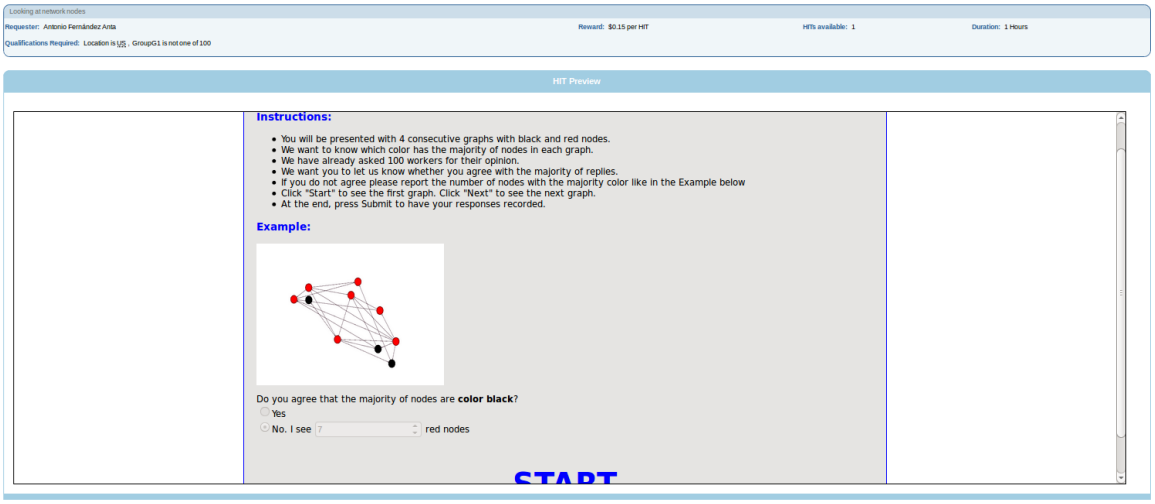


Figure 5.3: **HIT preview, for the task variation majority, as seen by the worker.**

Workers must: not have participated in our experiments in the past.

In this HIT we inform the worker that the majority of the received answers claims that nodes with a particular color form a majority in the graph. We ask the worker to let us know whether she agrees with the majority. If the worker disagrees, she must compute the number of nodes in the majority. In particular, in the difficult subtasks showing G1 and G2, the question claims that the majority has reported red as the majority color in the graph. While, in the easy subtasks showing graphs G3 and G4, the question claims the truth that the nodes in majority are of color black. Aside from discussion in the previous HIT about the difficulty of identifying the majority without counting, now counting the number of nodes with different colors can be confusing in the case of graphs in Figure 5.1 G1 and G2. Again, we do not provide any explanation to the workers as to the methodology we will use to reward them (based on the validity of all 4 replies or proportionally) for the reasons mentioned before. Workers are only aware that completing correctly a task, according to the requesters standards, will receive $0.15.

Count HIT Design

The initial preview of the HIT implementing the count variation is presented in Figure 5.4, while the HIT parameters are the following:

Title: Looking at network nodes;

Description: Let us know the number of nodes in a graph;

Keywords: graph, network, count;

Rewards per assignment: $0.20;

Number of assignments per HIT:100;

Time allotted per assignment: 1h;

HIT expires in: 7 days (with possibility of extending the initial expiration time);

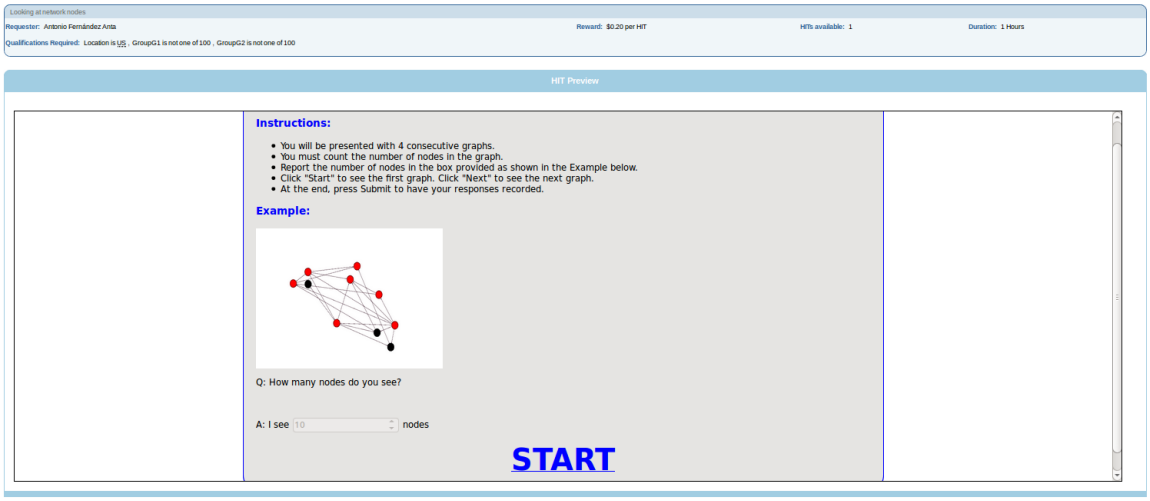


Figure 5.4: **HIT preview, for the task variation count, as seen by the worker.**

Auto-approve and pay Workers in: 7 days;

Require that Workers be Masters to do your HITs: No;

Workers must: be from USA;

Workers must: not have participated in our experiments in the past.

Again, the workers that accept to compute the task based on the HIT preview (see Figure 5.4) are presented with the four graphs of Figure 5.1 in a random order and are asked to count the number of nodes in the graph. The graphs G1 and G2 are obviously more difficult in comparison to G3 and G4 in Figure 5.1. No further explanations are provided to the workers as to the methodology used to reward them besides the fact that if the reported task result is marked correct they will receive $0.20.

5.2.2 Observations

In this subsection, after having received 100 responses for each of the three HITs, we analyze the data collected and we assess our hypothesis. Our first hypothesis is that regardless the difficulty of a subtask, a number of workers will respond correctly. Another hypothesis that we make is that for any given subtask, regardless of its difficulty, a number of workers will reply incorrectly. This hypothesis is actually contradicted by our collected data. To evaluate the correctness of a worker's response, we talk about a worker's accuracy which is the ratio of the sum of the worker's correct subtask answers over the total number of subtasks in a HIT. Thus, our third hypotheses claims that workers' accuracy and response time are correlated. Our fourth hypotheses claims that workers' accuracy is related with the task difficulty. Finally, we also conjecture that a number of workers reply with a guessed answer.

Hypothesis A: Given a subtask, regardless its difficulty a number of workers will reply correctly

To test this hypothesis we collected data from the three HITs for all subtasks. Figure 5.5 presents the number of workers giving a correct and an incorrect response for HIT color. Figure 5.6 shows a histogram of the number of workers giving a correct and an incorrect response for HIT majority. Finally, Figure 5.7 shows the number of workers counting correctly and incorrectly the nodes of the graphs.

As we can notice, even for graphs G1 and G2 that are the most difficult independently of the HIT question, there is a percentage of workers replying correctly. Even when we ask a precise question (i.e. HIT count) asking the workers to count the number of nodes a 10% and 29% of the workers for graphs G1 and G2 respectively reply with the correct answer.

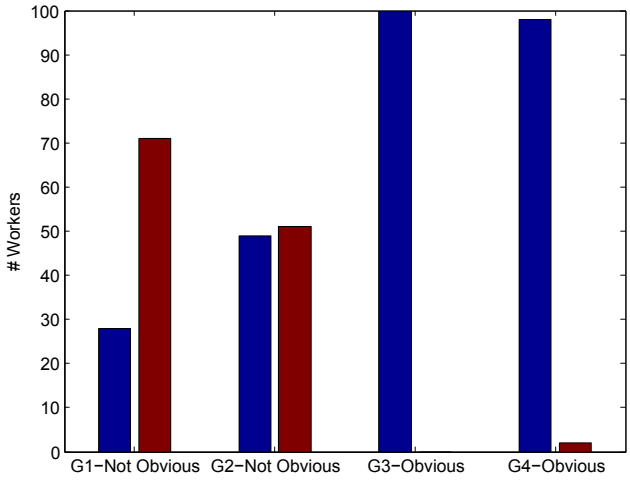


Figure 5.5: **Number of workers' correct and incorrect replies in all four subtasks for HIT color.** We present with the blue color (left) bar the correct replies and with the red color (right) bar the incorrect replies.

Hypothesis B: Given a subtask, regardless of its difficulty a number of workers will reply incorrectly

The data received from our experiments, as Figures 5.5 and 5.6 show, are actually contradicting our hypothesis. In the experiment HIT color and HIT majority, in G3 and G4 graphs respectively, all workers have reported the correct task result. We believe that the reason behind this is not the high reliability of the workers, something that is contradicted in the rest of the subtasks, but rather the almost obvious answer of the task combined with a 50% probability of randomly selecting the correct reply. This last argument is supported also by the fact that all workers replied correctly in different representations of the same graph in HIT color and HIT majority. Additionally, in both HITs two distinct workers replied incorrectly to the different representation

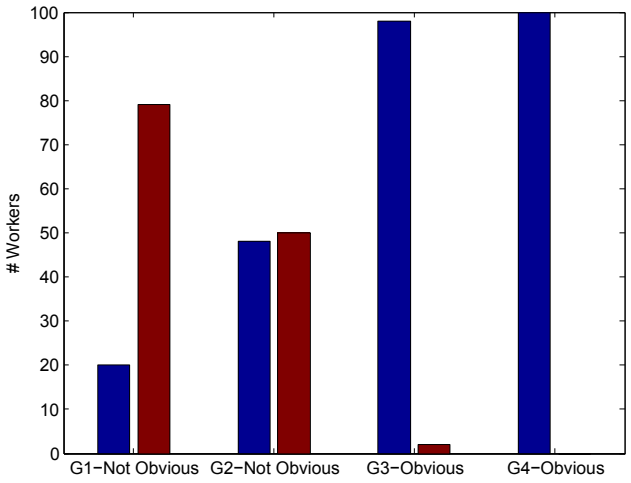


Figure 5.6: **Number of workers' correct and incorrect replies in all four subtasks for HIT majority.** We present with the blue color (left) bar the correct replies and with the red color (right) bar the incorrect replies.

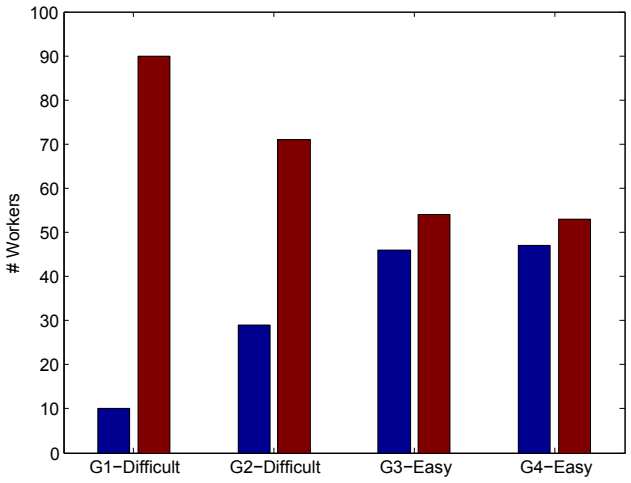


Figure 5.7: **Number of workers' correct and incorrect replies in all four subtasks for HIT count.** We present with the blue color (left) bar the correct replies and with the red color (right) bar the incorrect replies. In all four graphs the correct answer is equal to the mode.

of the same graph.

As we can notice, when the question is more explicit, like in HIT count where the exact number of nodes is asked, the majority of the workers reply incorrectly, even in the two easy subtasks showing graphs G3 and G4, which is certainly unexpected.

It is interesting to note that in HITs color and majority the percentage of workers reporting a correct result in G1 is much lower than 50% in comparison with G2 (different visualization of G1) where we almost received the same number of correct and incorrect responses. This is due

to the fact that graph G1 has somehow a non-intended optical illusion. If the worker does not devote enough time to analyze the graph she might easily believe that the majority color is red since red nodes are more concentrated in the middle of the graph and draw more attention. This is an indication that some workers choose not to devote time to verify a seemingly trivial answer.

Hypothesis C: Workers' accuracy and response time are correlated

We want to examine if there is any correlation among the time a worker needs to provide a reply and the correctness of the reply. For this reason we calculate the accuracy of each workers' answer as the ratio of the worker's correct subtask responses over the number of subtasks. As a first approach to test our hypothesis we check what is the linear correlation of the total response time of each worker with the worker's accuracy in all three HITs posted. These correlations are shown in Table 5.1.

Color	Majority	Count
0.2339	0.5024	0.3879

Table 5.1: **Correlation coefficient of the workers' response time with the accuracy of the workers (ratio of worker's correct subtask responses over all subtasks in the HIT.** Columns represent the three HIT tasks published on AMT.

As we can see from Table 5.1 there is a relationship among worker's accuracy and total worker response time, that is more apparent in HIT majority. Although an existing linear correlation among accuracy and response time affirms our hypothesis it does not give any further information, thus we want to examine better this relationship. To this end we will look at the Empirical Cumulative Distribution Function (ECDF) of the different accuracy groups in a HIT with the total workers' total response time to the HIT. We compute the ECDF according to the Kaplan-Meier estimate [86]. This estimator is usually used for survival or failure times data, that is the time a certain element of a study remained active after a treatment or the time a machine part needs to fail, etc. In our case we use this method to observe the time a worker needs to respond to four subtasks. In particular we categorize our workers in five different accuracy groups, based on the number of the correct subtask responses and we observe the response time of each accuracy group member. Hence, we have accuracy group zero where workers did not reply correctly to any of the subtasks; accuracy group one where workers belonging to that group replied correctly only to one subtask, and so on.

It looks like HIT majority is the most interesting case, and hence we will look first at the ECDFs of the total response time for accuracy groups two, three and four, depicted in Figure 5.8. As we see from Figure 5.6, accuracy group zero is not present while there are at most two workers in accuracy group one, which makes these two cases not interesting. The results of Figure 5.8 show a large difference between accuracy group two and accuracy group three and four. Combining these results with the ones of Figure 5.6, appears that almost all workers replied correctly to the two easy tasks. Some of the workers where able to give three correct responses, replying

correctly also to a difficult subtask, in particular 37 workers, while 15 workers replied correctly to all four subtasks. Notice that for accuracy groups three and four, only 10% of the workers reply within the first minute, in comparison with accuracy group two, where around 60% of the workers reply in the first minute to all four subtasks. This observation not only shows that workers' accuracy is correlated with the response time but also indicates that a worker's accuracy is correlated with the task difficulty. Later on, we revisit this last observation, discussing the workers' behavior.

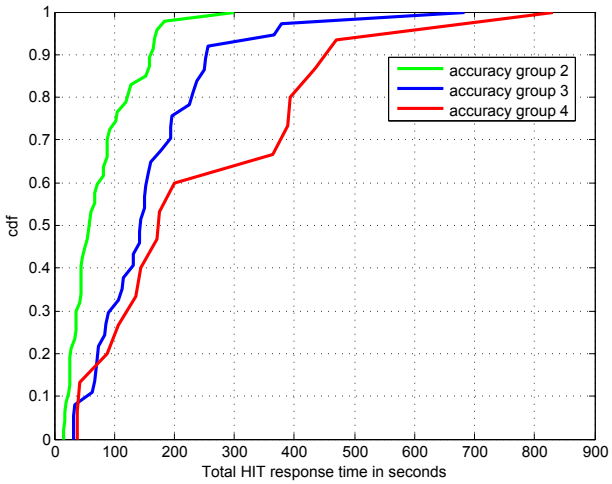


Figure 5.8: **The ECDF of the total response time for HIT majority.** From left to right, the workers with accuracy degree two up to degree four.

Moving on, we examine the ECDF of the total response time for the HIT color and each of the three accuracy groups (groups one and two are empty or have one-two elements), as shown in Figure 5.9. We observe that for HIT color for accuracy groups two and three there is a large correlation among time and accuracy. As we can see more than 90% of the workers in these groups reply within 1 minute, while it takes the 90% of the workers in accuracy group four more than 2.5 mins to reply. Notice that in all three accuracy groups a 25% of the workers replied within 30 seconds to all four subtasks. This observation suggests that a number of workers replied correctly simply because they guessed right, rather than counting for example the nodes. Comparing Figure 5.8 with Figure 5.9, we notice that, first of all, the distinction among accuracy group two and three, in terms of response time, is not so clear in HIT color again pointing to a guessing behavior from the side of the workers.

Finally, we look at the ECDF of the total response time for HIT count, shown in Figure 5.10. Notice that, in this case, where a more difficult task is asked of the workers, one where they can not guess the correct answer, accuracy depends even more on the response time. Workers in accuracy group four needed at least 5 minutes to provide their four correct replies. We can also notice that, there is a roughly constant separation in the plot among the five accuracy groups for values between 20% and 80%. This distance is likely to arise from the fact that accuracy of a

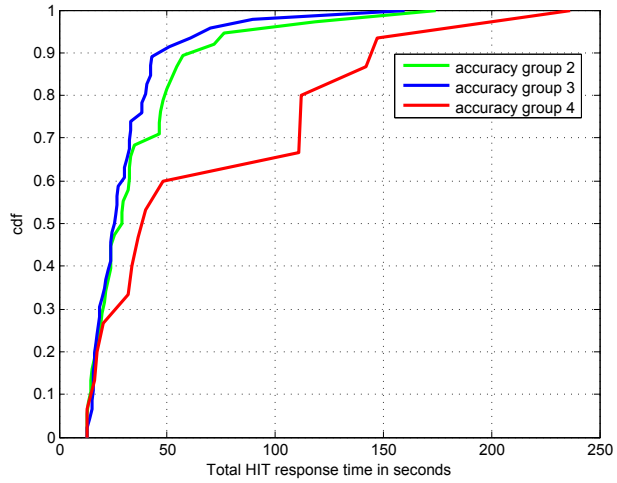


Figure 5.9: **The ECDF of the total response time for HIT color for the five accuracy groups.**

worker is correlated to the response time. Notice that for values more than 80% this correlation does not exist anymore. We believe that this is due to the fact that workers devote time in counting the nodes but they simply fail to provide the correct answer. On the other hand, for values below 10% is clear that workers in accuracy groups one, two and three might have guessed the correct reply.

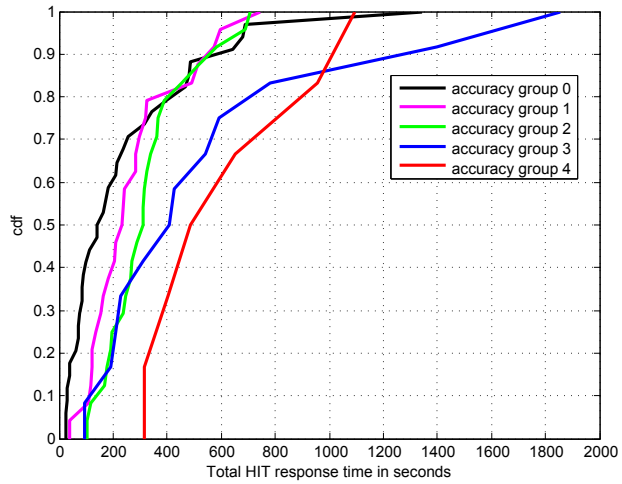


Figure 5.10: **The ECDF of the total response time for HIT count for the five accuracy groups.**

Hypothesis D: Worker's accuracy is related with the task difficulty

The three graphs in Figures 5.8- 5.10, indicated that there is a relationship among a worker's accuracy and the difficulty of a task. We have seen this relationship in all three tasks, each of which has 2 easy subtasks and 2 difficult subtasks. The difference among the three HITs is that

while color and majority HITs ask a multiple choice question, HIT count asks a question where the response is a positive integer, thus harder to guess.

In HIT color as we can see from Figure 5.9 and Figure 5.5 accuracy depends on the difficulty of the subtask. Subtasks with graphs G3 and G4 are easy to spot and thus all workers responded correctly. Conversely, for subtasks with graphs G1 and G2 it is more difficult for a worker to compute or guess the correct response, hence less workers reply correctly to them. This observation is also backed up by the response time as shown in Figure 5.9, some of the workers in accuracy group three either guessed for graphs G1 and G2 or devoted time only to one of the two graphs and thus the response time for group two and three is very similar.

Another way to see this relationship among accuracy and task difficulty, is through the number of workers belonging to each accuracy group for the three HITs, shown in Table 5.2. It is obvious that the difficulty of HIT count is affecting the accuracy of the workers. In addition, it seems that accuracy is not affected much by the different type of question asked in HIT majority. Of course, this observation does not provide enough information since the worker replying negatively to our question can still provide the wrong number of nodes in the majority. Figures 5.11 and Figures 5.12 show the answers of the workers that replied negatively to the requester's question in HIT majority. In the question regarding graph G1 only 25% of the workers that replied negatively found the correct answer, while in the question regarding G2 37.5% of the workers that replied negatively found the correct answer. If we compare these results with the results of Figure 5.13 and Figure 5.14, where we have the histogram of the workers, answer in HIT count, we can see that, in the case of graph G1 only 10% of the workers replied correctly while in the case of graph G2 29% of the workers replied correctly. Although in the case of the HIT majority, the sample is quite small, this is an indication that accuracy depends on the task difficulty, which in this case is counting only the black nodes instead of the whole set of nodes.

	Color	Majority	Count
Group zero	0	0	34
Group one	1	1	24
Group two	38	47	24
Group three	46	37	12
Group four	15	15	6

Table 5.2: **The number of workers belonging to each accuracy group in all three HITs.**

Conjecture: A number of workers reply with a guessed answer

It is clear from Figures 5.8- 5.10 that given the way a task is presented to the workers a "guessing" behavior is more or less likely to be observed. The average response time of accuracy groups two, three and four is less in the case of HIT count compared to the other two HITs. Also the average response time of accuracy groups two, three and four is larger in HIT count compared to the other two HITs. Thus, HIT color is more likely to provoke a guessing behavior from some

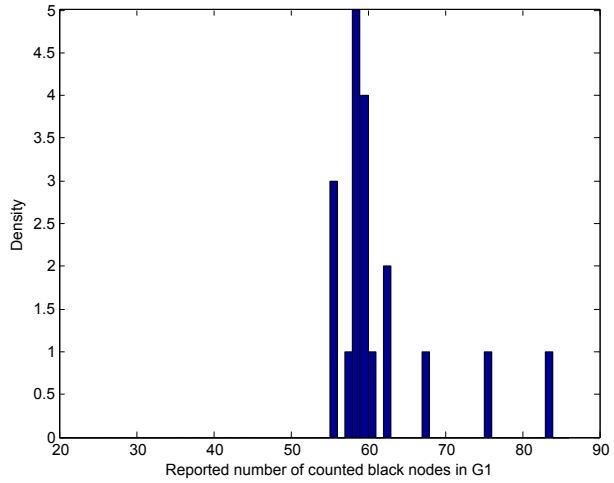


Figure 5.11: **Histogram of the density of reported number of black nodes in graph G1 for HIT majority.**

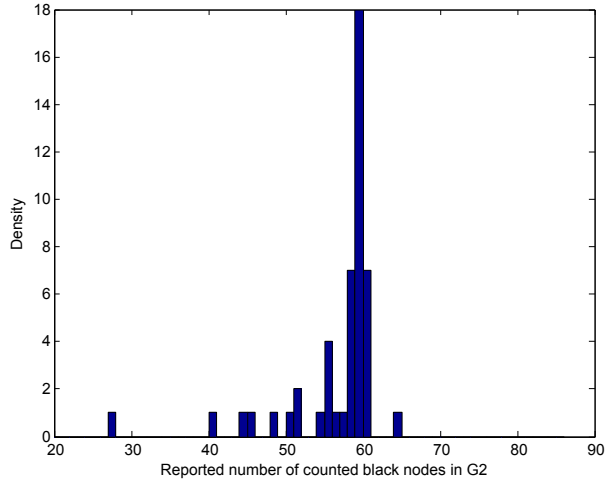


Figure 5.12: **Histogram of the density of reported number of black nodes in graph G2 for HIT majority.**

workers, HIT majority is a bit less likely to provoke a guessing behavior from the workers, while HIT count polarizes somewhat the behavior of the workers, by either forcing them to count or guess with high error probability.

As we have noticed before in Figure 5.9, 30% of the workers in all three accuracy groups examined respond with in 30 seconds in total. This is indicative of a guessing behavior from those workers, where some of them are able to guess correctly. Notice that 80% of the workers that had full accuracy, i.e., correct response ratio one, replied within 2 minutes. Even the faster worker in the case of HIT count that had full accuracy needed almost 5 mins to respond to the requester as we see in Figure 5.10, hence we can conclude that many workers in HIT color that have full

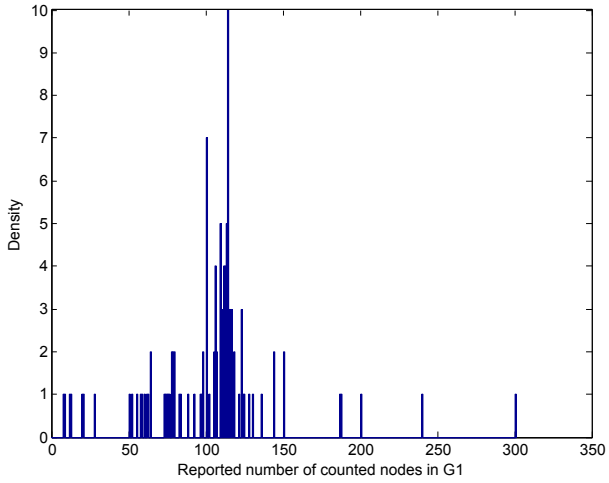


Figure 5.13: **Histogram of the density of reported number of nodes in graph G1 for HIT count.** The correct answer is 114.

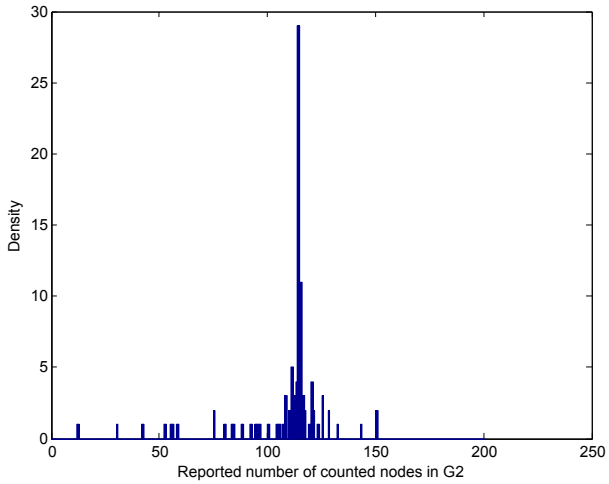


Figure 5.14: **Histogram of the density of reported number of nodes in graph G2 for HIT count.**The correct answer is 114.

accuracy did not count the nodes but rather devoted time in guessing the correct answer. We have similar observations for the case of HIT majority, but this time the percentage of workers that replied within 2 minutes correctly to all four subtasks is 30%. This is due to the fact that devoting time to guess correctly is not enough and more time is required to count the majority nodes.

Finally, the values depicted in Table 5.3 allow us to see in which graphs the workers are prone to guess. Table 5.3 shows the correlation coefficient values among the worker's response time in each graph and its correct response ratio (in the four subtasks). It is immediately obvious for graphs G3 and G4, that are easy graphs, and HITs color and majority, that are HITs that allow guessing, the correlation is very low. Thus, it is clear that highly accurate or not, workers are

	G1	G2	G3	G4
Color	0.2852	0.1945	0.0478	-0.0013
Majority	0.4905	0.3791	0.1270	0.2196
Count	0.4083	0.3009	0.4040	0.0969

Table 5.3: **The correlation coefficient of the worker's correct response ratio (in the four subtasks) with the response time in each graph.** Columns represent the correlation coefficient for each the graph and rows represent the HIT task.

responding in roughly the same time, that is very low as we have already observed. This is a clear indication of guessing the answer in these easy graphs. Another thing that we can notice from Table 5.3 is the high correlation in the difficult tasks G1 and G2 in HITs majority and count. Workers with high accuracy invest also a lot of time in these graphs, which makes us conclude that correct guessing behavior in these graphs is smaller.

5.3 Categorizing the workers' behavior

The observations of the previews section give us a comprehensive view of the workers' behavior. This behavior is consistent with the taxonomy of workers we have assumed in the modelling part of this work. That is, workers' behavior can be mapped to the following types: (1) altruistic with an error probability, (2) troll with an error probability, (3) pure altruistic, (4) pure malicious, and finally (5) rational workers.

We begin by studying the rational workers behavior in the experiments we conducted. One of our main conjectures from the experimental observations is that a number of workers reply with a guessed answer and for some workers this guessing behavior is induced by the question asked. In other words, some workers try to guess the correct answer because they have a high probability of identifying it without a significant cost. In our experiments, costs translate into time. We have observed a rational behavior from some workers in the HIT color where they maximize their benefit by trying to receive the payment while investing a minimal amount of time in the task. Comparing HIT color with HIT majority, where an extra amount of time is needed for the worker to provide the correct answers, can provide us with an additional indication of rationality. Recall that 80% of the workers that replied correctly to all four subtasks in HIT color replied within 2 minutes, while in the case of HIT majority only 30% of the workers that replied correctly to all four subtasks replied within 2 minutes. Thus, seems like a number of workers tried to maximize their benefit by providing a fast answer and hoping that they guessed correctly, while a larger number of workers tried to invest more time the task to get the correct answer and secure payment. In particular, in the HIT majority 80% of the workers belonging to accuracy group four replied within 6 minutes, investing 3 times longer in replying.

Identifying pure altruistic workers in our experiments is more easy. It is clear from the HIT count that workers replying to all four subtasks correctly and investing more than 10 minutes

to count the correct task result can be considered pure altruistic. By pure altruistic we refer to workers that would always reply correctly to a subtask. What separates these workers from rational workers that reply correctly is the fact that they invest an unreasonably large amount of time to respond to the requester. Unfortunately, we can not compute any percentage of altruistic behavior in our experiments since it is possible that workers with higher cognitive capabilities, thus faster in counting nodes, can be altruistic.

Again, through the HIT count we can identify pure malicious as the workers belonging in accuracy group zero. That is, they have not replied correctly in any subtask and they have done it in less than 1 minute. In a set of 34 workers, 20% of the workers have exhibited this behavior. It is possible that more workers have a pure malicious behavior and they are slower in giving a reply, but this is not clear as their behavior could also be characterized as the one of a worker aiming at replying fast and correctly, and failing.

We have observed that accuracy is related to the task difficulty. This observation supports the claim that workers might be acting in a altruistic way, aiming at reporting the correct answer but they fail due to the task difficulty, for example. In Figure 5.10 we noticed that it existed a case where a worker took 30 minutes to respond, and still she replied wrongly to one subtask. This is an extreme observation, that points out the existence of altruistic workers that suffer from errors.

Finally, the behavior of trolls with an error probability can be spotted in the workers that replied to more than one subtask correctly in roughly 30 seconds, in the HIT color and majority. In the HIT count is more difficult to spot such a behavior due to the nature of the task.

We noticed that given our task, it was more or less easy to spot some of the workers behaviors. These behaviors are mainly identified through the time a worker needs to respond to a subtask or to the whole task in comparison with its accuracy (i.e., the accuracy group it belongs in the HIT). To further check whether we have correctly identified the workers behavior we run a k-mean clustering algorithm [15] for each HIT with a vector of the workers response time and four parameters specifying whether the worker replied correctly or not to the subtask. The k-mean clustering algorithm we used here uses the squared Euclidean distance measure and the cluster k-means++ algorithm for cluster center initialization as it is implemented in MATLAB [15]. For each HIT vector we have run the k-mean algorithm evaluating the number of centroids. The way we choose the optimal centroid value was through visualization of the created clusters by k-mean. We visualized the data by plotting the scatter plot of the total response time against the number of correct responses in each task. We have found that while 2 centroids were giving two almost distinct clusters it was not enough to provide us with any useful information. On the other hand, 4 and 5 centroids were too many and there was no clear meaning in the clusters. Thus, we concluded that 3 centroids are enough to provide us with sufficient information, without having clusters interfering too much with each other.

In Figure 5.15 we can observe the k-mean clustering for three centroids, for the HIT color, for all four subtasks. In each subfigure we plot the accuracy group of the worker with the total response time in seconds. As can be observed, the three cluster groups created by k-mean algo-

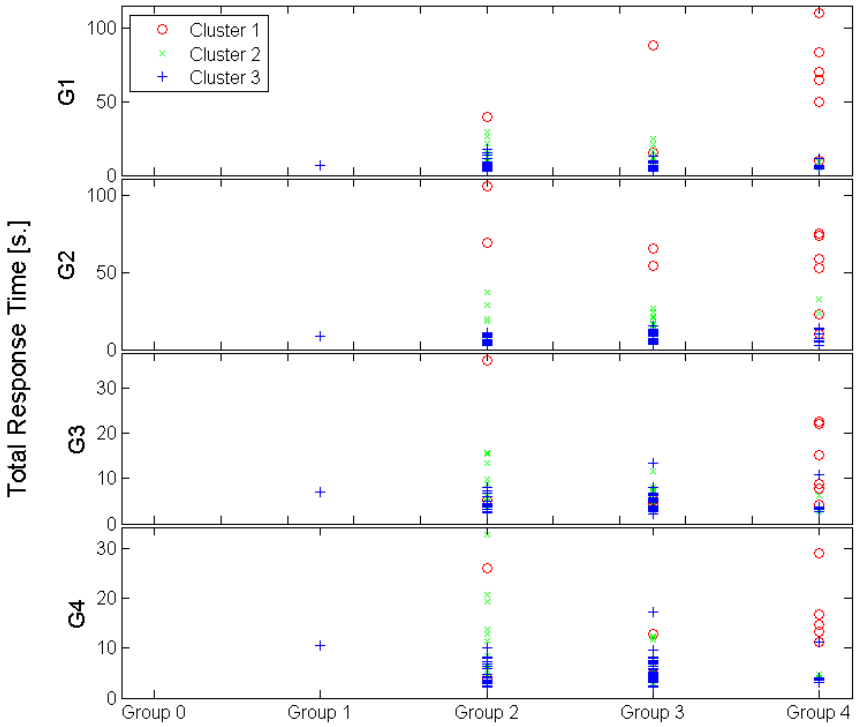


Figure 5.15: **Scatter plot of the workers' response time against workers accuracy group for all four HIT color subtasks.**

rithm do form visible groups, although there are points where there is not a clear visual separation among the clusters. The scatter plots of Figure 5.15 are consistent with our observation that accuracy is related to response time. From the way the clusters are formed we could conclude that cluster 1 could represent altruistic workers that have an error probability, while cluster 3 could represent troll workers with an error probability. This reasoning on the workers behavior can also be justified by the scatter plot of Figure 5.16 and Figure 5.17.

Rationality in Figure 5.15, Figure 5.16 and Figure 5.17 can be associated with cluster 2. Independently of the worker's correct response ratio the workers in cluster 2 devote on average the same amount of time across subtasks of the same difficulty. Notice that the amount of workers in cluster two with accuracy four is significantly less while workers in cluster two are almost evenly partitioned among the rest of the possible accuracy groups. This observation indicates that workers in cluster 2 have a threshold on the amount of time devoted to a task that does not depend on their accuracy. Thus, generally speaking, workers of cluster 2 can be characterized as rationals.

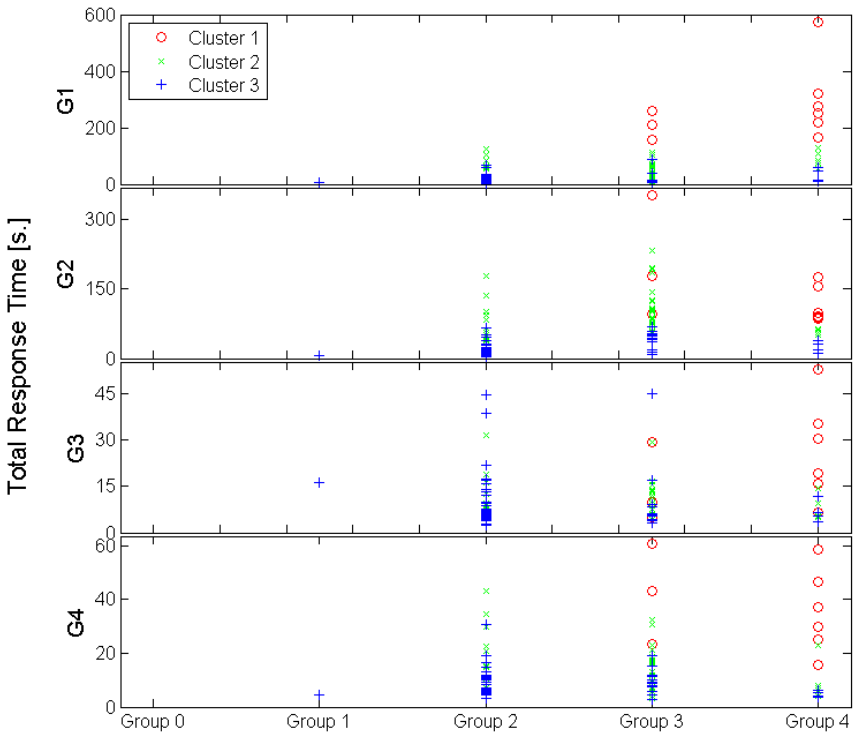


Figure 5.16: **Scatter plot of the workers' response time against workers accuracy group for all four HIT majority subtasks.**

5.4 Conclusions

This chapter summarizes our main findings on the behavior of the workers on AMT when three HITs with different computational complexities are each computed by 100 workers. Our HITs are designed in such a way to capture the relationship among workers' accuracy with the task difficulty and the time invested in the task. Indeed, we have clearly seen that more than 50% of the workers reply with an incorrect answer to a complex to compute subtask, regardless of the complexity of the question asked (i.e. find the majority, count etc.). A bit to our surprise, we have also noticed that if the subtask complexity and also the question complexity is low, that is a binary set of answers is presented, in two cases all 100 workers replied correctly. Thus, one of the general observation was that a worker's accuracy is correlated with the task difficulty. Another general observation we took is that a worker's accuracy is correlated with the response time to the task. Finally, we conjecture that a number of workers is actually guessing the correct answer rather than accurately calculating it, something that become obvious when the workers were asked to compute the precise number of nodes in the presented graphs.

Our observations regarding the relationship of the worker's accuracy with regards to the task

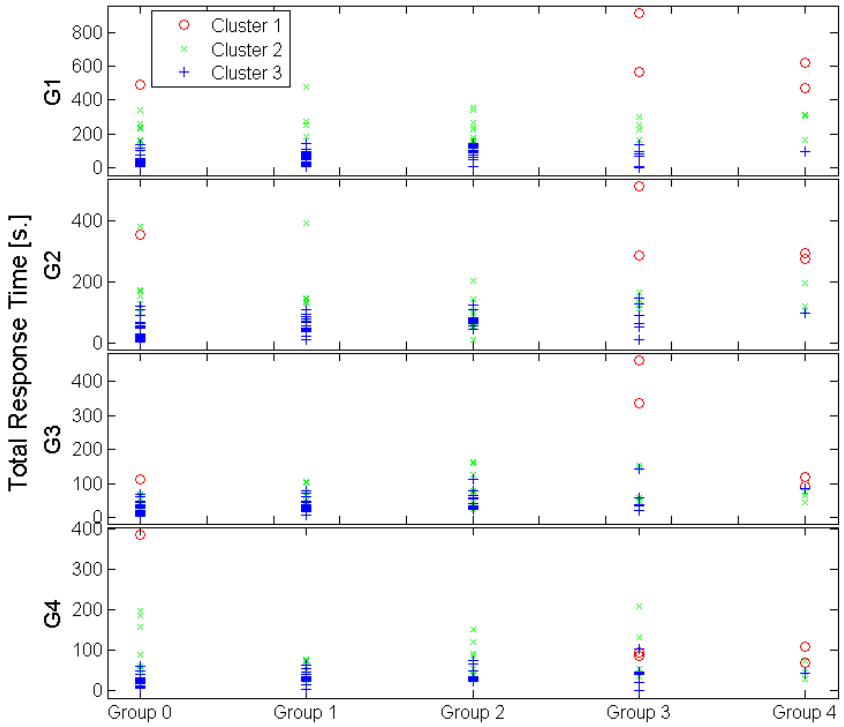


Figure 5.17: **Scatter plot of the workers' response time against workers accuracy group for all four HIT majority subtasks.**

difficulty and the time invested are actually agreeing with the literature on the topic. This actually supports the validity of our experiments, but the main contribution of this work comes from the fact that through our particular observations we are able to draw conclusions that support our beliefs on the workers' behavior. In particular, we have significant data that support the existence of workers behaving according to the rational model or viewed from another perspective workers that are altruistic but suffer from errors. Moreover, we have also some indications that could justify a model assuming the presence of pure malicious and pure altruistic workers or another model that assumes the presence of troll workers with an error probability. Gadiraju et al. [54] conducted experiments on CrowdFlower investigating the workers behavior on crowdsourced surveys. From their findings, among other types of worker they were able to spot fast deceivers and smart deceivers, these two types they observed are similar to the rational behavior we observed in our experiements. Moreover, Gadiraju et al. observed a type of worker which they call gold standard preys. This type of worker has a similar behavior to altruistic workers with an error probability. In their work, Gadiraju et al. classified workers into five different types according to the received responses and on the workers responses to auditing questions, while in our work the derived conclusions are based on the response time and the subtask difficulty in combination with

the received response.

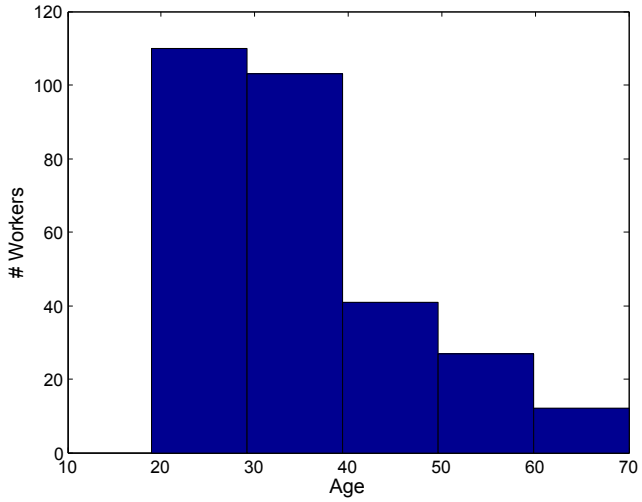


Figure 5.18: **The reported age of participating workers in all three HITs**

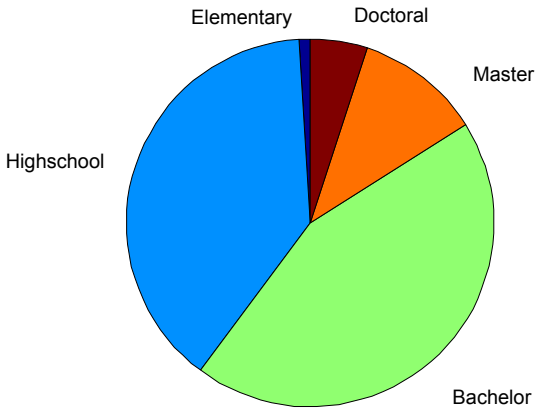


Figure 5.19: **The reported education of participating workers in all three HITs**

Finally, we conclude our observations in this chapter with some demographics, which were collected after the workers have replied to the four subtasks of each HIT. We have mentioned to the workers that replying to these demographics questions was not obligatory. As we have seen 47.6% of the worker population reported they are women, while 51.3% reported they are men and the rest of the worker choose not to reply. Figure 5.18 shows that the majority of workers are among 20-40 years old. In Figure 5.19 we can observe that more than half of the workers have received a higher education while in Figure 5.20 we see that a bit more than half of the workers don't have a fixed occupation.

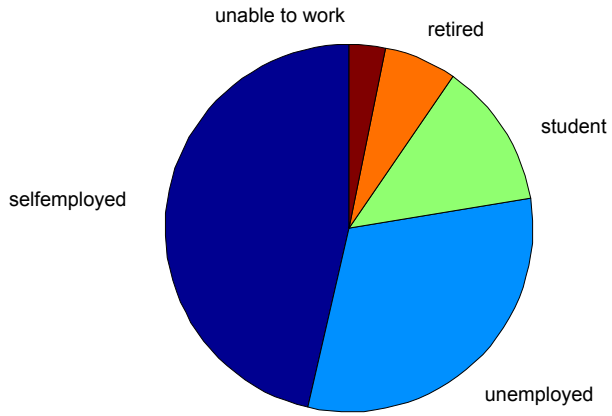


Figure 5.20: **The reported occupation of participating workers in all three HITs**

Chapter 6

Fair and Efficient Distribution of Resources

6.1 Introduction

In the previous chapters we have addressed the problem of reliability in Internet-based task computing systems. In Section 4.5 we have presented a mechanism for achieving reliability where the master, in each round, selects the most reliable workers to take part in the computation. Up to now, we have abstracted our problem considering the presence of only one master and thus focusing on the untrustworthy nature of the workers. But, in all the examples of Internet-based computing such as volunteer computing and crowdsourcing, we have multiple masters besides multiple workers. In the case of volunteer computing, workers can participate in multiple volunteer projects supported by Berkeley Open-source software for volunteer computing (BOINC). While in crowdsourcing platforms such as Amazon Mechanical Turk (AMT), masters are competing for the same workers, paying the same fee rate to the platform. Imagine a scenario where a master keeps assigning tasks to a worker that is considered highly reputable not only by her but also by other masters, such that the worker does not accept to compute other masters' tasks in that time interval. This is a case where one master is monopolizing a valuable common "resource", while all masters have equal rights to be sharing that resource. Thus the question rises, how can the masters share the reputable workers in a fair manner?

This problem could be trivially solved if the reputable workers where computing tasks for the master in a round-robin fashion. Such a solution is not efficient though, since master might not get to assign a task to a worker when she values it the most. Thus, another approach would be that a worker is assigned to the master that values her contribution the most. In this solution, masters exactly like workers can be rational and thus declaring a different valuation, than their actual one, on the reputable workers, in an attempt to increase their benefit.

A number of mechanism have been designed dealing with the problem of allocating resources in the presence of strategic agents, for example auction mechanisms. The two most famous auc-

tion mechanisms are the Vickrey-Clark-Groves (VCG) auction [16] and the Generalized Second Price (GSP) auction [46]. In both mechanisms, payment is present for incentivizing the correct behavior of the participating agents according to the mechanism designer's goal. In the volunteer computing and crowdsoucing example, it is infeasible to have payments from the side of the masters. Considering the first example, payments would defeat the concept of volunteer computing and as for the crowdsourcing example masters expect to have fair access to the workers since they are already paying a fee rate for it.

Given all the above, in this chapter we solve the problem at hand by designing a mechanism for Fair and Efficient Distribution of Resources (FEDoR) , in the presence of strategic agents and without the use of monetary incentives. There is a plethora a real world applications deriving from computer science or from social sciences were payments can not be used as incentives in resource allocation problems. For example allocating Central Processing Unit (CPU) cycles or sharing of valuable hospital equipment. For this reason we choose to present our mechanism in a general manner, talking about strategic agents instead of masters that share a set of indivisible goods or resources instead of making reference to workers in particular.

Resource allocation in an efficient and fair manner among strategic agents (we use the terms agent and player interchangeably) that can misreport their values to increase their benefit, is a non-trivial problem. The most straightforward approach is the design of mechanisms where, once the solution concept is achieved, fairness and efficiency are guaranteed. In order for the designed mechanism to be incentive compatible, economic incentives/payments must usually be designed. In this work, we do not use any economic incentives/payments and hence our mechanism design goals are achieved without any incentive constraint, that is without money.

Based on the nature of the problem, we assume a Bayesian setting where agents have private information. In the master-worker paradigm, the masters' private information is a function of the reputation value assigned to every worker in each round. Agents can be cheating, declaring preferences that do not correspond to their true valuations in an effort to increase their utility. That is a master might continually reporting that she has high valuation on a worker while this might not be the case. Unlike mechanisms with payments, were money incentives enforce the good behavior of the agents, this is not the case here. FEDoR is designed especially to detect cheating and punish it. This is feasible since FEDoR is designed to work in a repeated setting, which allows the mechanism to gain knowledge on the agents' valuations and appropriately punish misbehaving agents, incentivizing their good behavior. The mechanism is able to guarantee fairness and social efficiency by exploiting the concept of linking mechanisms [71] described in Chapter 2. That is when a lot of independent copies of the same decision problem are linked together, then no incentive constrains are needed for agents to be truthful. We are assuming that preferences are declared within a common unit measure. In realistic applications, valuations might not even appear within the same unit of measure. Agents might not be able to associate their valuation under a common unit of measure. To deal with this matter FEDoR uses a common normalization process called Probability Integral Transformation (PIT) [13]. Using PIT any cumulative probability distribution

function can be transformed to the uniform [0,1] distribution. Essentially, the PIT of a value x extracted from a probability distribution is the aggregated probability of the values no larger than x in the distribution. Intuitively, it is similar to the percentile of x. Under the assumption that FEDoR has a historical sample of the agents' valuations, a distribution fitting the agents valuations can be defined. Since knowing the agent's valuation distribution is the only thing that we need to know to apply the PIT, from this point onwards we will assume that the valuations declared by the agents follow the uniform [0,1] distribution. In the work by Santos et al. [117], this approach was used in an analogous way to transform the agent's cost to the uniform distribution (see Fig 1 in [117]). Having transformed the agent's valuation to the uniform [0,1] distribution, a Goodness of Fit (GoF) test can define whether an agent's valuation is following its true distribution, and act in an analogous manner; this guarantees that FEDoR is a truthful mechanism (as we prove below).

Contributions

We design a mechanism for Fair and Efficient Distribution of Resources (FEDoR) in the presence of strategic agents. We consider a multiple-instances, Bayesian setting, where in each round the preference of an agent over the set of resources is a private information. We assume that in each of r rounds n agents are competing for k non-identical indivisible goods, ($n > k$). In each round the strategic agents declare how much they value receiving any of the goods in the specific round. The agent declaring the highest valuation receives the good with the highest value, the agent with the second highest valuation receives the second highest valued good, etc. Hence we assume a decision function that assigns goods to agents based on their valuations. The novelty of the mechanism is that no payment scheme is required to achieve truthfulness in a setting with rational/strategic agents. The FEDoR mechanism takes advantage of the repeated nature of the framework, and through a statistical test is able to punish the misreporting agents and be fair, truthful, and socially efficient. The mechanism is fair in the sense that, in expectation over the course of the rounds, all agents will receive the same good the same amount of times. Through FEDoR , we solve the problem of allocating workers to masters in an Internet-based computing platform in a fair and efficient manner without payments. Masters in different rounds have different valuation on the workers based on the private reputation value they have for each worker. Thus, we guarantee that in expectation over the course of the computation rounds, all masters will receive each worker the same amount of times when they value them the most (based on the reputation value each master has on the workers). Moreover, we perform comparison with two benchmark auction mechanisms presenting the advantage of FEDoR, in Internet-based task computing systems.

Besides allocation of workers in volunteer computing and crowdsourcing platforms, FEDoR is an eligible candidate for other applications that require fair distribution of resources over time. For example, equal share of bandwidth for nodes through the same point of access. But further on, FEDoR can be applied in less trivial settings like sponsored search, where payment is necessary

and can be given in the form of a flat participation fee. FEDoR can be a good candidate in a setting like that to solve the problem of starvation of publicity slots for some advertisers that have a difficult time determining their true valuations.

The rest of the Chapter is structured as follows, in Section 6.2 we present a general model, instead of focusing in the master-worker paradigm. In Section 6.3 we present our designed mechanism and in Section 6.4 we show that our mechanism satisfies the properties of fairness, truthfulness and social efficiency. Section 6.5 presents simulations comparing FEDoR to VCG and GSP in Internet-based task computing systems. Finally, in Section 6.5 we provide an extensive discussion on other feasible applications of FEDoR.

6.2 Model

Before moving any further, let us define the setting we are considering as well as the problem we are solving.

Definition 6.1 (Setting)**.** *We consider the presence of n risk-neutral players, $N = \{1, 2, \ldots, n\}$. Every one of the n players participates in a (a priori unknown) number of consecutive instances (each instance is considered a round) of an allocation game. In each instance of the allocation game:*

1. *k heterogeneous non-divisible goods are allocated among the n players, where $1 \leq k < n$.*
2. *For every good i of the k goods of the round there exist a relative value depicted by the weight w_i (goods in each round can be different but the relative values remain the same). We assume that goods are sorted by decreasing weight, i.e., $w_1 \geq w_2 \geq \cdots \geq w_k$.*
3. *Players declare how much they value obtaining any of the goods of the set. This is expressed by a single value.*

The outcome set of the game played is the set $\mathcal{D}$ of all k-permutations of N (i.e., all posible permutations of the subsets of N of size k). Hence, the outcome $d = (d_1, d_2, \ldots, d_k) \in \mathcal{D}$ is the ordered list of players to whom the goods will be allocated, so that the player d_i will receive the ith good (whose weight is w_i).

Each player *privately* observes her preferences over the alternatives in $\mathcal{D}$ before the collective choice in the game is made. This is modeled by assuming that player i privately observes a parameter θ_i, which determines her preferences over obtaining a good from the set. We say that θ_i is the player type, for every player i. The set of all possible types of player i is Θ_i. We denote by $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$ the vector of player types. The set of all possible vectors is $\Theta = \Theta_1 \times \Theta_2 \times \ldots \Theta_n$. We denote by θ_{-i} the vector obtained by removing θ_i from θ.

Thus, we denote by $\Pi = \Delta(\Theta)$ (in general, we denote by $\Delta(S)$ the set of all probability distributions over some set S) the set of all probability distributions over Θ. It is assumed that

there is a common prior distribution $\pi \in \Pi$ that is shared by all the players. We denote by $\pi_i \in \Delta(\Theta_i)$ the marginal probability of θ_i. We define $\pi_i(\theta_i|\theta_{-i})$ as the conditional probability distribution of θ_i given θ_{-i}.

In this work we assume that the player types are normalized, in the sense that $\Theta_i = [0, 1]$, for every player i. Moreover, for every player i the marginal probability π_i of θ_i is also normalized as the uniform continuous distribution in the interval $[0, 1]$. This are not simple assumptions as in the work by Santos et al. [117], applying the PIT we can transform any probability distribution to the uniform [0,1] Finally, we assume that all distributions π_i are independent, i.e., $\pi(\cdot|\theta_{-i}) = \pi_i(\cdot)$. For this reasons we have made the simplification of having our notations free from the concept of the round.

Players' preferences over outcomes are represented by a utility function $u_i(d, \theta_i) \in \mathbb{R}$ defined over all $d \in \mathcal{D}$ and $\theta_i \in \Theta_i$. In this work the utility function is defined as $u_i(d, \theta_i) = \theta_i \sum_{j=1}^{k} w_j \delta_{id_j}, \forall i \in N, \forall d \in \mathcal{D}$, where δ_{ab} is the Kronecker delta.

We assume that the set of outcomes $\mathcal{D}$, the set of players N, the type sets in Θ, the common prior distribution $\pi \in \Pi$, and the utility functions u_i are *common knowledge* for all the players. Similarly, the game rules defined by the mechanism used are also common knowledge. However, the specific type θ_i observed by player i belongs to her *private information*.

A strategy for the player i is a map $\sigma_i : \Theta_i \to \Delta(\Theta_i)$, where $\sigma_i(\hat{\theta}_i|\theta_i)$ is the conditional probability that the player reports $\hat{\theta}_i$ when her true type is θ_i. A strategy σ_i is *truthful* (and we say that the player is *honest*) if, for every pair $(\hat{\theta}_i, \theta_i)$, $\sigma_i(\hat{\theta}_i|\theta_i) = 1$ if $\hat{\theta}_i = \theta_i$ and 0 otherwise. As usually done, we will use $\hat{\theta}_i$ to denote the reported type, and θ_i the actual type.

We assume the availability of a *GoF* test, that can test whether a value (a reported type in our case) is a sample from a uniform distribution in $[0, 1]$. Hence, $GoF\_Test(\hat{\theta}_i)$ is true if and only if $\hat{\theta}_i$ is a uniform random sample. By this definition we assume that the GoF test is perfect, but this is a theoretical artifice. In Section 6.5, through simulations we find a good approximation to a perfect GoF test.

In this work, we search for a mechanism $\langle \Theta, g \rangle$, where $g : \hat{\Theta} \to \Delta(\mathcal{D})$ is the decision function. With this function, $g(d|\hat{\theta})$ is the conditional probability that the mechanism decides $d \in \mathcal{D}$ when the players report $\hat{\theta}$. The mechanism must be without utility transfers (payments) and satisfy the following properties.

- *Fairness.* Every player i (honest or not) gets the jth good, the same proportion of times in expectation, for every j. I.e.,

$$\mathrm{E}_{\theta,\hat{\theta}\in\Theta}\left[\sum_{d\in D:d_j=i} g(d|\hat{\theta})\right] = \mathrm{E}_{\theta,\hat{\theta}\in\Theta}\left[\sum_{d\in D:d_j=i'} g(d|\hat{\theta})\right], \forall i, i' \in N, \forall j \in [1, k], \forall \sigma. \tag{6.1}$$

- *Truthfulness.* The strategy that maximizes the utility of a player i is to be honest. I.e.,

$$\mathrm{E}_{\theta,\hat{\theta}\in\Theta}\left[\sum_{d\in\mathcal{D}} u_i(d,\theta_i)g(d|\theta_i,\hat{\theta}_{-i})\right] \geq \mathrm{E}_{\theta,\hat{\theta}\in\Theta}\left[\sum_{d\in\mathcal{D}} u_i(d,\theta_i)g(d|\hat{\theta})\right], \forall i \in N, \forall\sigma. \quad (6.2)$$

- *Social Efficiency.* If all players are honest, the expected social utility with decision function g is maximized with respect to any other decision function g'. I.e.,

$$\mathrm{E}_{\theta\in\Theta}\left[\sum_{i\in N}\sum_{d\in\mathcal{D}} u_i(d,\theta_i)g(d|\theta)\right] \geq \mathrm{E}_{\theta\in\Theta}\left[\sum_{i\in N}\sum_{d\in\mathcal{D}} u_i(d,\theta_i)g'(d|\theta)\right]. \quad (6.3)$$

6.3 The FEDoR Mechanism

FEDoR solves the problem of allocating multiple resources in a multi-instance setting, with a decision function that is such that the properties of fairness, truthfulness and social efficiency are satisfied without utility transfers. Algorithm 13 describes the steps that a generic player i executes and the actions taken by the mechanism. This algorithm represents how the mechanism can be applied in a distributed setting, in the absence of a central authority that takes the allocation decision.

Algorithm 13 FEDoR Algorithm (code for player i)

1 **foreach** k *set of goods* **do**
2 *Observe type* θ_i
3 *Choose type* $\hat{\theta}_i$ *using probability distribution* $\sigma_i(\cdot|\theta_i)$ (*player i's strategy*)
4 *Broadcast the type* $\hat{\theta}_i$ (*this is the bid of player* i)
5 *Wait* **to receive** *the reported types* $\hat{\theta}$ *from all the players in* N
6 $d \leftarrow g_f(\hat{\theta})$ \\ *applying the FEDoR mechanism*
7 **if** $(i = d_j)$ **then** *player* i *receives the* j*th good*

In every instance of the allocation game (defined as a new announcement of k goods to be allocated) a player i observes its type θ_i. Then, applying its strategy σ_i, the player chooses a type $\hat{\theta}_i$ that will be reported as the player's bid. These bids are gossiped among all players, so that all of them end with the same vector of reported types $\hat{\theta}$. We are assuming that communication among players is perfect (reliable and synchronous). Once a player has received all the reported types from the rest of the players, the algorithm moves on, applying the mechanism.

Each player applies the mechanism to obtain the allocation decision. The decision function $g_f : \Theta \rightarrow D$ of FEDoR is one where $g_f(\hat{\theta}) = (i_1, i_2, \ldots i_k)$ such that $(i_1, i_2, \ldots i_k) : v_{i_1} \geq v_{i_2} \geq$

$\dots v_{i_k}$ and $v_{i_k} \geq v_i \forall i \in N \setminus \{i_1, ..i_k\}$ where

$$v_i = \begin{cases} \hat{\theta}_i & \text{GoFtest}(\hat{\theta}_i) \\ \text{pseudorandom}(\hat{\theta}_{-i}) & \text{otherwise} \end{cases}$$

In other words the decision function of FEDoR is one that given the vector of declared types $\hat{\theta}_i$ provides the vector of players $(i_1, i_2, \dots i_k)$ receiving the k goods, by i_j we declare the player receiving the jth good. The decision of the mechanism is based on a cheaters (following a strategy besides declaring their true type) detection scheme. Recall that, as we explained in Section 6.2 each player has a uniform continuous distribution in the interval [0,1] over all instances. This is a common information that all players share and allows them to verify announced types of the rest of the players. If the player i declared type $\hat{\theta}_i$ passes the GoF test then the declared value is accepted. Otherwise the declared value is replaced, by each player, by a pseudorandom value uniform in $[0, 1]$ derived from the rest of reported types $\hat{\theta}_{-i}$. Since all players apply the same perfect GoF test to the same value $\hat{\theta}_i$, and the same pseudorandom function to the same vector $\hat{\theta}_{-i}$, they all assign the same value to v_i. As a result, the same vector v is obtained by each player, which contains the values of the players that passed the test and the generated values for those that did not. The mechanism allocates the k goods in order to the k players with the highest value in the vector v.

It is important to notice that the algorithm is the same for all players and that it is based on information known by all of them. Therefore, no central entity is required to apply the FEDoR mechanism. Of course, if the environment is such that a central authority is available, the algorithm can trivially be transformed to accommodate a centralized solution using FEDoR.

6.4 Formal Analysis

In this section we analyze the FEDoR mechanism and we show that the three desired properties described in Section 6.2, namely fairness, truthfulness, and social efficiency are satisfied. We start by proving the latter.

We show that FEDoR is socially efficient if all players are honest. To prove this, first we prove that there is no mechanism that has a decision function that gives a higher utility to a player, independently of the declared types Θ.

Lemma 6.1. *There is no mechanism $\langle \Theta, g \rangle$ such that*

$$E[\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g(d|\theta)] > E[\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g_f(d|\theta)].$$

Proof: Assume the claim is not true. Then, there is a mechanism $\langle \Theta, g \rangle$ such that

$$E[\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g(d|\theta)] > E[\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g_f(d|\theta)]. \tag{6.4}$$

This means that there is at least one θ such that

$$\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g(d|\theta) > \sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g_f(d|\theta). \tag{6.5}$$

Let $d_x = (x_1, x_2, \ldots, x_k)$ be an outcome that maximizes $\sum_{i \in N} u_i(d_x, \theta_i)$.
Then, $\sum_{i \in N} u_i(d_x, \theta_i) \geq \sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g(d|\theta)$. On the other hand, let $g_f(d_f|\theta) = 1$ for $d_f = (i_1, i_2, \ldots, i_k)$. By the definition of u_i, this means that $\sum_{j=1}^{k} \theta_{x_j} w_j > \sum_{j=1}^{k} \theta_{i_j} w_j$. However, this is not possible, since $w_1 \geq w_2 \geq \cdots \geq w_k$, and the values θ_{i_j} are the largest values in θ in decreasing order, i.e., $\theta_{i_1} \geq \theta_{i_2} \geq \cdots \geq \theta_{i_k}$. Hence, the mechanism $\langle \Theta, g \rangle$ does not exist. ■

Since there is no mechanism that has a decision function that is better than the one of FEDoR mechanism, then we can also proof that when all players are honest (a.k.a declaring their true type) the social utility (i.e. the total utility of all players) is maximized.

Theorem 6.1 (Social efficiency)**.** *For any mechanism $\langle \Theta, g \rangle$ and strategies $\sigma_i, \forall i \in N$, it holds that*

$$E[\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g_f(d|\theta)] \geq E[\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g(d|\hat{\theta})|\sigma_1, \ldots, \sigma_n].$$

Proof: Assume the claim is not true, and hence there is a mechanism $\langle \Theta, g \rangle$ and strategies $\sigma_i, \forall i \in N$, such that

$$E[\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g_f(d|\theta)] < E[\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g(d|\hat{\theta})|\sigma_1, \ldots, \sigma_n]. \tag{6.6}$$

Then, let us define a new mechanism $\langle \Theta, g' \rangle$ as follows. $g'(d|\theta) = \int_0^1 \int_0^1 \ldots \int_0^1 g(d|\hat{\theta}) \cdot \sigma_1(\hat{\theta}_1|\theta_1) \cdot \ldots \cdot \sigma_n(\hat{\theta}_n|\theta_n) d\hat{\theta}_n d\hat{\theta}_{n-1} \ldots d\hat{\theta}_1$. The decision function $g'(\cdot)$ assigns the same probability to each possible outcome d as the combination of the strategies $\sigma_i(\cdot)$ and the decision function $g(\cdot)$. Thus,

$$E[\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g_f(d|\theta)] < E[\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g(d|\hat{\theta})|\sigma_1, \ldots, \sigma_n] \tag{6.7}$$

$$= E[\sum_{i \in N} \sum_{d \in \mathcal{D}} u_i(d, \theta_i) g'(d|\theta)], \tag{6.8}$$

contradicting Lemma 6.1. ■

We will now start proving the truthfulness property of the FEDoR mechanism. Let us denote by $E[\hat{U}_i|\sigma_1, \ldots, \sigma_n]$ the expected utility of player i under the FEDoR mechanism when players follow strategy $\sigma_i, \forall i \in N$.

Lemma 6.2. *For any non empty set of players $S \subset N$, the total expected utility of the players in S does not depend on the strategies of the rest of players.*

Proof: We start by proving that the expected utility of a player i does not depend on the strategy of another player $h \neq i$. In particular, we will show that $E[\hat{U}_i|\sigma_x, \forall x \in N] = E[\hat{U}_i|\sigma_x, \forall x \neq h; h \text{ honest}]$, i.e., that the utility of i does not depend on whether h is honest or not, which implies the former statement.

For every player $x \in N$, let $f_x(v_x, \theta_x)$ be the density function of the pairs of values (v_x, θ_x), where θ_x is the type observed by player x and v_x is the entry corresponding to x in the vector v used to decide in FEDoR (see Algorithm 13). If the strategy $\sigma_x(\cdot)$ induces a uniform distribution on the reported types $\hat{\theta}_x$ (recall that θ_x does follow a uniform distribution), then the values v_x will always be the declared types (bid) $\hat{\theta}_x$. Otherwise, the GoF test will always fail and v_x will be pseudorandom values generated from the rest of declared values. In any case, the values v_x are uniformly distributed, and the function f_x is independent from the rest of functions $f_i, \forall i \neq x$. Moreover, the marginal distributions of $f_x(v_x, \theta_x)$ must also be uniform.

Let us consider any player $i \in S$. The expected utility $E[\hat{U}_i|\sigma_x, \forall x \in N]$ of the player can then be computed as

$$\sum_{d\in\mathcal{D}} \underbrace{\int_0^1 \cdots \int_0^1}_{2n} u_i(d|\theta_i) g_f(d|v) f_1(v_1,\theta_1)\cdots f_n(v_n,\theta_n) \tag{6.9}$$

$$dv_1 \cdots dv_n d\theta_1 \cdots d\theta_n \tag{6.10}$$

$$= \sum_{d\in\mathcal{D}} \underbrace{\int_0^1 \cdots \int_0^1}_{2n-1} u_i(d|\theta_i) g_f(d|v) f_1(v_1,\theta_1)\cdots \tag{6.11}$$

$$f_{h-1}(v_{h-1},\theta_{h-1}) f_{h+1}(v_{h+1},\theta_{h+1})\cdots f_n(v_n,\theta_n) \tag{6.12}$$

$$\int_0^1 f_h(v_h,\theta_h) d\theta_h dv_1 \cdots dv_n d\theta_1 \cdots d\theta_{h-1} d\theta_{h+1} \cdots d\theta_n \tag{6.13}$$

$$= \sum_{d\in\mathcal{D}} \underbrace{\int_0^1 \cdots \int_0^1}_{2n-1} u_i(d|\theta_i) g_f(d|v) f_1(v_1,\theta_1)\cdots \tag{6.14}$$

$$f_{h-1}(v_{h-1},\theta_{h-1}) f_{h+1}(v_{h+1},\theta_{h+1})\cdots f_n(v_n,\theta_n) \tag{6.15}$$

$$dv_1 \cdots dv_n d\theta_1 \cdots d\theta_{h-1} d\theta_{h+1} \cdots d\theta_n. \tag{6.16}$$

The first equality follows from the independence of f_h from the other f_j. The last equality follows from the fact that $\int_0^1 f_h(v_h,\theta_h)d\theta_h = 1$, from the uniform marginal distribution of f_h with respect to θ_h. Now, applying a change of variable, we can replace in the above expression

v_h by θ_h, resulting after reordering in

$$E[\hat{U}_i|\sigma_x, \forall x \in N] = \sum_{d\in\mathcal{D}} \underbrace{\int_0^1 \cdots \int_0^1}_{2n-1} u_i(d|\theta_i) g_f(d|v_{-h}, \theta_h) \tag{6.17}$$

$$f_1(v_1,\theta_1)\cdots f_{h-1}(v_{h-1},\theta_{h-1}) f_{h+1}(v_{h+1},\theta_{h+1})\cdots \tag{6.18}$$

$$f_n(v_n,\theta_n) dv_1 \cdots dv_{h-1} d\theta_h dv_{h+1} \cdots dv_n d\theta_1 \cdots d\theta_n \tag{6.19}$$

$$= E[\hat{U}_i|\sigma_x, \forall x \neq h; h \text{ honest}]. \tag{6.20}$$

This property can now be applied iteratively for every player $h \notin S$, showing that the expected utility of $i \in S$ is independent of the strategy of the players in S. Summing over all the players in S the claim of the lemma is proved. ■

Using Lemma 6.2 we can proof the truthfulness property.

Theorem 6.2 (Truthfulness)**.** *The strategy that maximizes the utility of a player i is to be honest. I.e.,* $E[\hat{U}_i|\sigma_x, \forall x \in N] \leq E[\hat{U}_i|\sigma_x, \forall x \neq i; i \text{ honest}]$.

Proof: Assume for contradiction that the claim is not true. Then, there are strategies $\sigma_x, \forall x \in N$ such that

$$E[\hat{U}_i|\sigma_x, \forall x \in N] > E[\hat{U}_i|\sigma_x, \forall x \neq i; i \text{ honest}]. \tag{6.21}$$

From Lemma 6.2 we have the following identities (the third one uses linearity of expectations).

$$E[\hat{U}_i|\sigma_x, \forall x \in N] = E[\hat{U}_i|\sigma_i; x \text{ is honest}, \forall x \neq i] \tag{6.22}$$

$$E[\hat{U}_i|\sigma_x, \forall x \neq i; i \text{ honest}] = E[\hat{U}_i|\text{all players are honest}] \tag{6.23}$$

$$E[\sum_{h\neq i} \hat{U}_h|\sigma_i; x \text{ is honest}, \forall x \neq i] = E[\sum_{h\neq i} \hat{U}_h|\text{all players are honest}] \tag{6.24}$$

Replacing equations 6.22 and 6.23 in 6.21, we obtain

$$E[\hat{U}_i|\sigma_i; x \text{ is honest}, \forall x \neq i] > E[\hat{U}_i|\text{all players are honest}]. \tag{6.25}$$

Which added to 6.24 yields

$$E[\sum_{h\in N} \hat{U}_h|\sigma_i; x \text{ is honest}, \forall x \neq i] > E[\sum_{h\in N} \hat{U}_h|\text{all players are honest}]. \tag{6.26}$$

However, this contradicts Theorem 6.1. ■

We concentrate now in the fairness property. Recall the FEDoR mechanism as it was described in the previous subsection. Either because the player reports types that are uniformly distributed, or because they are replaced by FEDoR with uniform pseudorandom values. The values v_i of the vector v used to distribute the goods are independent random samples form a uniform distribution $[0, 1]$. Hence the following theorem.

Theorem 6.3 (Fairness)**.** *Every player i (honest or not) has the same probability $1/n$ of getting the jth good, for every j.*

Proof: Since the elements of the vector v are independent uniform samples as mentioned, the probability that the value v_i is the jth largest in v is $\left(\begin{array}{c} n-1 \\ j-1 \end{array} \right) (1-v_i)^{j-1} v_i^{n-j}$. Then the probability that a player i has the jth largest value in v, given that v_i takes values uniformly between 0 and 1 is

$$\int_0^1 \left(\begin{array}{c} n-1 \\ j-1 \end{array} \right) (1-v_i)^{j-1} v_i^{n-j} dv_i = \frac{1}{n} \tag{6.27}$$

■

Finally, we will analyze the expected utility of the players depending on their strategy. We start by presenting the expect utility of an honest player. Observe that the utility obtained is independent from the strategies of the rest of players, as proven in Lemma 6.2.

Theorem 6.4. *The expected utility of an honest player i is $E[\hat{U}_i|\sigma_x, \forall x \neq i; i \text{ honest}] = \frac{\sum_{j=1}^{k} w_j(n-j+1)}{n(n+1)}$. This value is independent from the strategies $\sigma_x, \forall x \neq i$.*

Proof: Since i is honest, she reports her true type θ_i, which follows a uniform distribution. Hence, it holds that $v_i = \theta_i$. As in the proof of Theorem 6.3, the probability that the type θ_i is the jth largest value is $\left(\begin{array}{c} n-1 \\ j-1 \end{array} \right) (1-\theta_i)^{j-1} \theta_i^{n-j}$. Then, the expected utility can be computed as

$$\begin{aligned} & E[\hat{U}_i|\sigma_x, \forall x \neq i; i \text{ honest}] \\ = & \sum_{j=1}^{k} w_j \int_0^1 \theta_i \left(\begin{array}{c} n-1 \\ j-1 \end{array} \right) (1-\theta_i)^{j-1} \theta_i^{n-j} d\theta_i \\ = & \frac{\sum_{i=1}^{j} w_j(n-j+1)}{n(n+1)} \end{aligned} \tag{6.28}$$

■

We now find the utility of a player i that is not honest (a.k.a cheater), but she reports values that are not uniform (and hence the GoF test always fails) or she reports types $\hat{\theta}_i$ that are uniform but independent of her true types θ_i. Observe that this theorem does not consider the case when the types reported are uniform and somehow correlated with the real types (this is left to be shown experimentally).

Theorem 6.5. *The expected utility of a dishonest player i that reports non uniform types or types independent of her true normalized uniform distribution is $E[\hat{U}_i|\sigma_x, \forall x \in N] = \frac{1}{2n} \sum_{j=1}^{k} w_j$.*

Proof: In this case, the value v_i used to distribute the goods follow a uniform distribution that

is independent of the actual type θ_i of player i. Hence,

$$\begin{aligned} & E[\hat{U}_i|\sigma_x, \forall x \in N] \\ = \sum_{j=1}^{k} w_j \int_0^1 \theta_i \int_0^1 \left(\begin{array}{c} n-1 \\ j-1 \end{array} \right) (1-v_i)^{j-1} v_i^{n-j} dv_i d\theta_i & \\ & = \frac{1}{2n} \sum_{j=1}^{k} w_j. \end{aligned} \tag{6.29}$$

■

6.5 Simulation Results

FEDoR is a mechanism for distributing/allocating goods, without monetary incentives, in the presence of strategic agents. It guarantees an allocation of the goods that will be fair and socially efficient, given that the mechanism is truthful (strategy-proof). The good properties of the mechanism are feasible under a specific setting, discussed in Section 6.2. As we mentioned in the introductory section, FEDoR is a good match for volunteer computing where no payments are involved and the crowdsourcing example where masters' expect a fair treatment since they pay a fee to the platform. For the sake of experimentation though, we would like to simulate FEDoR and compare it with VCG [16] and GSP [46] auction mechanisms. As far as we know, currently, no computational environment that corresponds to the Internet-based task computing system model uses auctions to assign workers to the masters, but this could be implemented in the future. Our goal is to study whether our mechanism behaves well in a scenario where payments are involved in comparison with well established auction mechanisms. On a side note, through this study we also examine the feasibility of auction mechanisms in Internet-based task computing systems for distributing the workers.

Simulations compare FEDoR with two benchmark auction mechanisms, VCG and GSP mechanisms in two cases: (1) only honest players are present, (2) dishonest players are present. Additionally, simulations examine the cost of not having a perfect GoF test and the length of the historical test that would yield an almost perfect GoF test. We choose to compare FEDoR with VCG that is a powerful strategy proof and efficient (in every instance) mechanism. In VCG the ith highest bidder gets the ith best good, but pays the *externality* that she imposes on the other bidders by winning that good. Besides its good properties, VCG is difficult to be understood by the bidders. Thus, we also compare FEDoR with GSP, a simplification of the previous mentioned mechanism that lack some essential properties. In GSP the ith highest bidder gets the ith best good, but pays to the seller the bid of the $(i+1)$st highest bidder. This mechanism is neither strategy proof nor efficient but is easily understandable by the bidder.

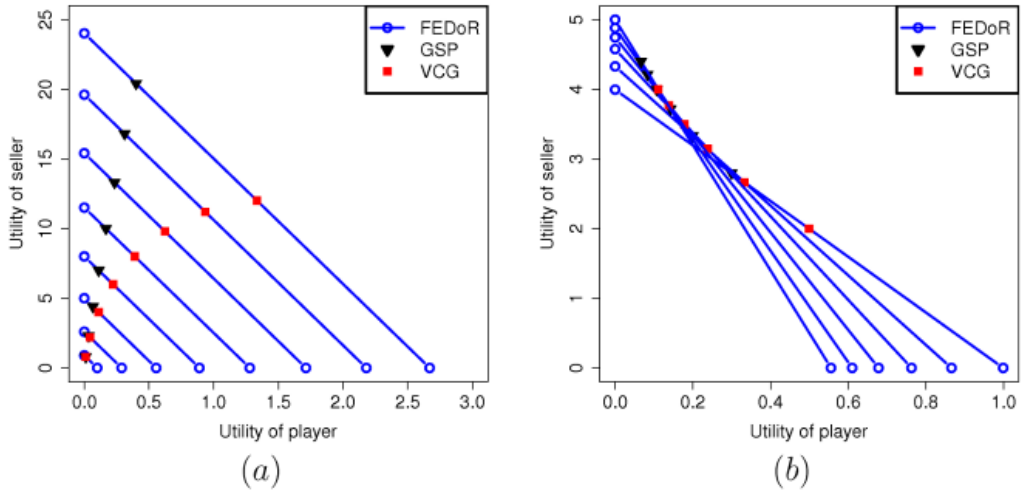
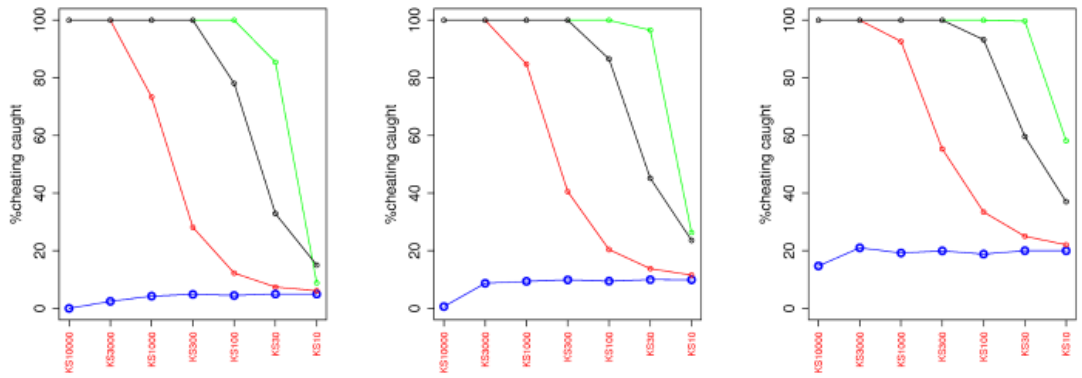


Figure 6.1: **Comparing FEDoR with VCG and GSP in the mean utility of seller (computational platform) and player (master) per auction.** The values for FEDoR form a line in each experiment, since the utilities of platform and masters can be tuned with the flat fee. The values for VCG and GSP always lie on the corresponding FEDoR line. Plots represent the mean utility per auction of the platform and one master over 10000 rounds of execution and 100 experiments. (a) Scenarios with 9 masters and the number of workers increasing from 1 (leftmost line) to 8 (rightmost line). (b) Scenarios with 3 workers and the masters decreasing from 9 (line with largest slope) to 4 (line with smallest slope).

Presence of only honest players: In this scenario we look at an instance where the masters are following an honest strategy. This could happen either because the mechanism is strategy proof, which is the case of VCG and FEDoR or because they just choose to follow an honest strategy. We consider the scenario of Internet-based task computing and we compute the utilities of the masters and the computational platform in all three mechanisms. The utilities are computed in two distinct scenarios: (a) the presence of 9 masters is assumed and the number of workers for which they compete k varies from 1 to 8; the weight for each worker values in decreasing order from k to 1. (b) the number of workers is fixed to $k = 3$ and the number of masters varies from 4 to 9. Additionally we assume that, for the purpose of experimentation, the GoF test used is perfect. The results obtained are presented in Fig 6.1, where we plot the utility achieved by one of the masters (all masters are honest and follow a uniform distribution) against the utility of the computational platform. In the case of FEDoR, we plot the utility values assuming that the same flat fee is paid by every master, so that the utility of the platform is simply this flat fee multiplied by the number of masters. The utility of a masters, on the other hand, is the value she assigns to the workers she gets (which is essentially the value given in Theorem 6.4) minus the flat fee.

In Fig 6.1(a) it can be observed that the utilities are the same for VCG (with externality) and GSP in the case of auctioning one good. When the number of workers, for which the masters



(a) (b) (c)

Figure 6.2: **The percentage of positives with the KS test as a function of the history length of the KS test.** (a) The upper bound threshold of the p.value of the KS test is 0.05, (b) the upper bound threshold of the p.value of the KS test is 0.1, (c) the upper bound threshold of the p.value of the KS test is 0.2. From bottom to top the lines represent the behavior in a scenario with no cheaters, cheaters bidding values with a beta distribution where $\beta = 0.9$, cheaters bidding values with a beta distribution where $\beta = 0.7$, and cheaters bidding values with a normal distribution $(\mu = 0.5, \sigma = 0.15)$.

are competing, increases, the utility of the platform is greater with GSP than with VCG, while on the other hand the utility of the master is greater with VCG than with GSP. This behavior was expected since GSP is a mechanism that favours the entity that applies the mechanism. Moreover, it is not strategy proof, something that would not put the master in an advantageous position if she is honest compared to VCG. The utilities with the FEDoR mechanism, on its hand, form a line that is a function of the flat fee the masters pay. It is worth to observe that all the points that correspond to GSP and VCG utilities are on the FEDoR line, which means that with the appropriate value of the flat fee, FEDoR can achieve the same platform and worker utilities as GSP and VCG. Moreover, the advantage of FEDoR, as it is shown in Fig 6.1(a), is that the flat fee provides a tradeoff between platform and master utilities, and allows to chose any point in the lines shown in the figure. Hence, FEDoR has an adjustable performance that can be changed depending on the needs of the platform. In Fig 6.1(b) we make similar observations and derive the same conclusions. Again the utilities of GSP and VCG lie on the FEDoR line, and the utilities with FEDoR can be tuned with the flat fee. In addition, we notice that when the number of masters increases, their utility decreases in all three mechanisms. On the contrary, the utility of the platform increases in all three mechanisms when the number of masters increases.

Presence of dishonest players: In this section we assume that the masters are dishonest, we set as our GoF test the Kolmogorov-Smirnov (KS) test and we evaluate its performance. Through simulations we examine the conditions under which the KS test performs optimally and we eval-

uate the impact of this optimality to our mechanism. Simulations show that FEDoR is a truthful mechanism. We compare the utility of FEDoR with VCG and GSP in a number of scenarios where the cheating behavior of the masters varies. Having as a base case the scenario where all masters are honest, we compare the way in which the honest and dishonest masters' utility is affected. Finally, we show how the social utility is affected by the presence of dishonest masters in all three mechanisms considered.

Unless otherwise stated, our results have been obtained by running 100 experiments of 10000 rounds (auctions) each. The number of participating masters is 9 and the number of workers for which they compete is $k = 3$. The weights assigned to the workers are $w_1 = 3, w_2 = 2, w_3 = 1$. Recall that honest masters are considered the ones that reveal their true valuation for the set of goods. Cheater masters are the ones declaring a different valuation from their actual valuation on the set of workers. All masters' private valuations follow independent uniform distributions in the interval $[0, 1]$.

To evaluate the impact of the presence of dishonest masters in their utility, the utility of the honest masters, and also in the social utility, we have simulated 10 distinct scenarios. The scenarios we consider are as follows. Scenario A: 9 honest masters following the uniform distribution, B: 8 honest masters and one cheater with normal distribution ($\mu = 0.5, \sigma = 0.15$), C: 8 honest masters and one cheater with beta distribution with $\beta = 0.9$, D: 8 honest masters and one cheater with beta distribution with $\beta = 0.7$, E: 8 honest masters and one cheater with a random uniform distribution (different from its own distribution of values), F: 6 honest masters and 3 cheaters with random uniform distributions, G: 6 honest masters and 3 cheaters with beta distributions ($\beta = 0.9$), H: 6 honest masters and 3 cheaters with beta distributions ($\beta = 0.7$), I: 6 honest masters and 3 cheaters with normal distributions ($\mu = 0.5, \sigma = 0.15$), J: 5 honest masters, 1 cheater with random uniform distribution, 1 cheater with beta distribution ($\beta = 0.9$), 1 cheater with beta distribution ($\beta = 0.7$), and 1 cheater with normal distribution ($\mu = 0.5, \sigma = 0.15$).

First we examine the performance of the KS test when it is used as the GoF test in FEDoR. Fig 6.2 evaluates the impact of the history length in the accuracy of the test. As it can be seen the KS test is a good fit for our mechanism. The history length refers to the number of values the test keeps as a reference to decide whether the new value reported by the master follows the uniform distribution. Notice that, in all three graphs of Fig 6.2, in the case where the master has a uniform distribution, as the history length gets smaller, the number of false positives increases. In addition to that, we have checked what happens in the case when the master cheats with a distribution that may resemble (or not) the uniform distribution. We have chosen the beta distributions with different parameter and a normal distribution. As you can see from the plots in Figs 6.3 and 6.4, with a beta distribution and a large β parameter, the master can approximate the uniform behavior, and thus it is difficult for the KS test to identify a cheater. Notice that in all three graphs of Fig 6.2, as the length of the history test decreases, the percentage of true positives decreases. For different values on the upper bound threshold of the p.value (which decides the acceptance of a value based on the history), we have noticed that using a $p.value < 0.1$ is a good balance

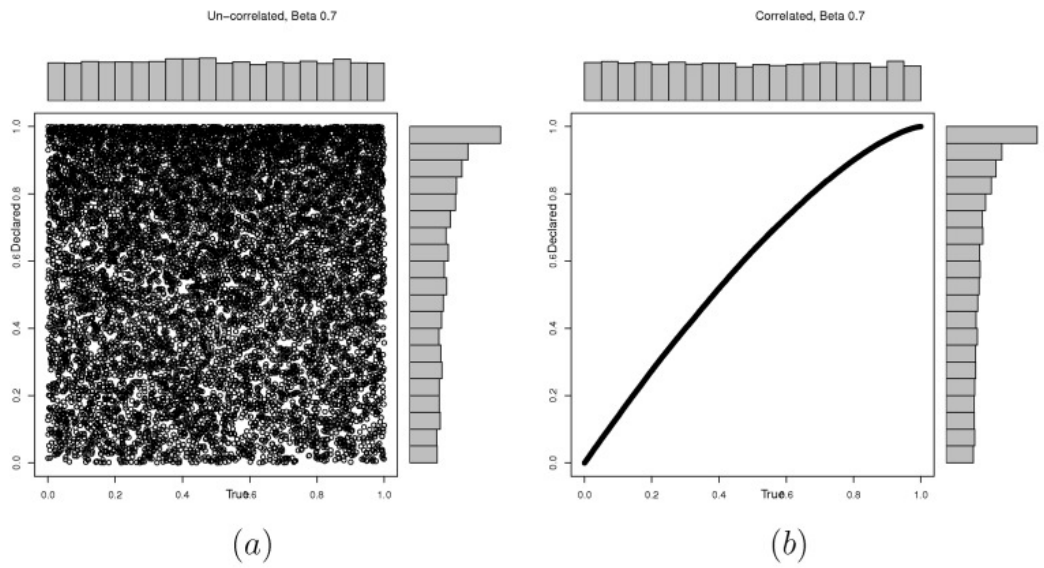


Figure 6.3: **Different strategies, modelled as a $\beta = 0.7$ probability distribution.** (a) The declared values are not correlated with the true values (following a uniform distribution). (b) The declared values are correlated with the true values (following a uniform distribution). Note that the marginal distribution represented in the x-axis corresponds to the declared values of the agent, while the y-axis represents the true values of the agent.

between having large percentage of true positives and not having too many false positives. For example, for a history length of 1000 rounds (which is a value that does not give a great overhead to the mechanism), in Fig 6.2(a), where $p.value < 0.05$, the true positives are close to 75%; in Fig 6.2(b), where $p.value < 0.1$, the true positives are close to 82%; while in Fig 6.2(c), where the $p.value < 0.2$, the true positives are close to 95% while the false positive are much higher than in the other scenarios (close to 20%). Hence, selecting 1000 rounds as the history of the mechanism, provides a good performance and is large enough if we compare it with the 10000 rounds that our scenarios execute as part of the online auction process. In the rest of the section we will run experiments with a history length of 1000 and KS test with $p.value < 0.1$. Before actual auctions are run, the history buffer is filled with 1000 values, so the results are not affected by history's transient states.

We investigate now the utility of the honest masters and cheaters in a variety of different scenarios, while having the KS test as our GoF test. In this study we assume that the masters pay no flat fee. We have shown analytically (assuming a perfect GoF test) that the best strategy for each masters is to be honest. Our experimental results come to assert this even in the case where the GoF is not perfect. From Fig 6.5(a), the utility of the honest master is greater than the cheater in all scenarios considered. Independently of the behavior of the rest of the masters, the mean utility of the honest master is around the same value of 5400. There is no cheating strategy that

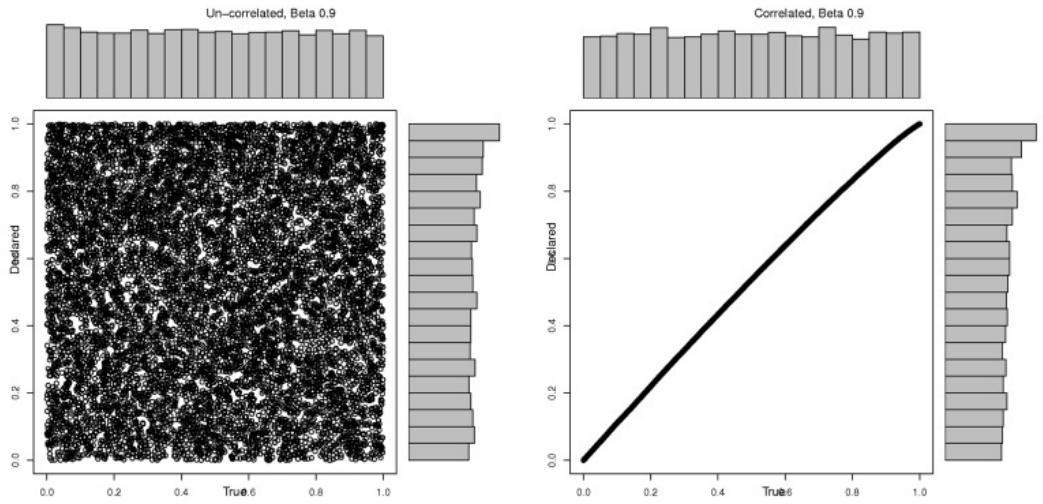


Figure 6.4: **Different strategies, modelled as a $\beta = 0.9$ probability distribution.**(a) The declared values are not correlated with the true values (following a uniform distribution). (b) The declared values are correlated with the true values (following a uniform distribution). Note that the marginal distribution represented in the x-axis corresponds to the declared values of the agent, while the y-axis represents the true values of the agent.

will give a higher utility. Especially in scenarios C and G where the master cheats with a beta (where parameter is $\beta = 0.9$) indeed the cheaters increase their mean utility by roughly around 500 units but still the mean utility of the honest master is around 1500 units higher. This also proves the efficiency of the KS test for $p.value < 0.1$ and history length 1000. The horizontal (blue) line in Fig 6.5(a) represents the value of utility as it is calculated from the analytical part. Notice that the analytical value is quite close to the experimental with the single exception of the master cheating with beta, where $\beta = 0.9$.

Going one step further now we compare the utility of masters when FEDoR is used with the cases where VCG and GSP are used. As Fig 6.5 shows, the honest master has always a larger utility compared with the utility of the cheating master in all scenarios considered, and in all mechanisms. However, in some scenarios the distance between the utility of a cheater and an honest master is very small (see, e.g., scenario H in Fig 6.5(b)). It is also interesting to notice that in FEDoR in scenario A, where all masters are honest, the utility of the master is maximized compared to the honest masters in the rest of the scenarios. This is not the case for the other two mechanisms (see scenario I for both VCG and GSP).

The utility of an honest master with VCG and GSP can vary significantly depending on the scenario. In any case, it is always around 1100 for VCG and 700 for GSP. Comparing with FEDoR, the utility of an honest master with the latter is up to four times larger than those with

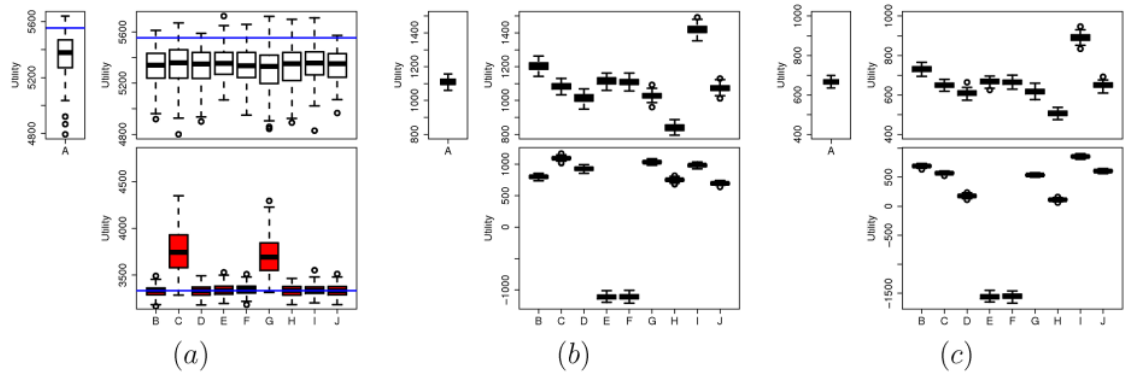


Figure 6.5: **The utility of an honest master compared to a cheating master in different scenarios.** (a) Using the FEDoR mechanism, for p.value$<$ 0.1 and history length 1000 (the blue line represents the analytical utility values). (b) Using the VCG mechanism. (c) Using the GSP mechanism. The left box for each mechanism (marked A) represents the distribution of the utility of an honest master in a scenario where all 9 masters are honest following a uniform. Right top box represents the utility of honest masters that follow a uniform distribution in scenarios $B-J$. Left bottom box represents the utilities of cheating masters with the distributions defined in scenarios $B-J$. Scenarios are as follows, B: 8 honest masters and one cheater with normal distribution ($\mu = 0.5, \sigma = 0.15$), C: 8 honest masters and one cheater with beta distribution with $\beta = 0.9$, D: 8 honest masters and one cheater with beta distribution with $\beta = 0.7$, E: 8 honest masters and one cheater with a random uniform distribution (different from its own distribution of values), F: 6 honest masters and 3 cheaters with random uniform distributions, G: 6 honest masters and 3 cheaters with beta distributions ($\beta = 0.9$), H: 6 honest masters and 3 cheaters with beta distributions ($\beta = 0.7$), I: 6 honest masters and 3 cheaters with normal distributions ($\mu = 0.5, \sigma = 0.15$), J: 5 honest masters, 1 cheater with random uniform distribution, 1 cheater with beta distribution ($\beta = 0.9$), 1 cheater with beta distribution ($\beta = 0.7$), and 1 cheater with normal distribution ($\mu = 0.5, \sigma = 0.15$), which is the one represented in the plot.

VCG and GSP. This is due to the fact that in our experiments we assumed that the flat fee of the master is zero. As we showed though in Fig 6.1, the flat fee is something that can be adjusted to match the desired utility of the master (or the platform).

Finally, from Fig 6.6 (a) we see that the social utility is maximized in FEDoR when all masters are truthful (scenario A compared with the rest of scenarios). This does not hold for the other two mechanisms. As you can see from Fig 6.6 (b) and (c) scenarios B and I (where dishonest masters cheat with normal distribution $\mu = 0.5, \sigma = 0.15$) have greater social utility than scenario A where they are all honest. We can also notice that the social utility in FEDoR is up to four times larger than those of GSP and VCG (but, of course, we are assuming no flat fee).

6.6 Discussion

Although powerful mechanisms (such as VCG) exist that, under monetary incentives, achieve the designed goals, cashing out payments is not always feasible. Maybe the system is distributed

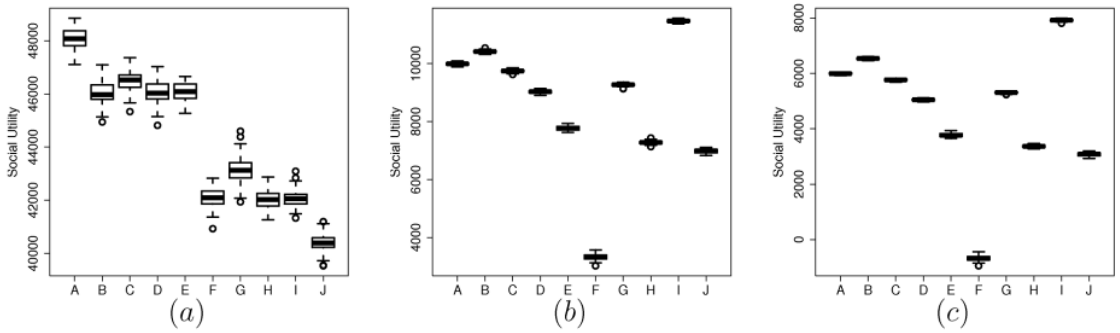


Figure 6.6: **The social utility in different scenarios.** (a) Using the FEDoR mechanism, for p.value$<$0.1 and history lenght 1000 (b) Using the VCG mechanism. (c) Using the GSP mechanism. The scenarios A to J are the same as in Fig 6.5.

and no central authority exists that can guarantee the payments, or payments are too impractical. Another possibility is that the nature of the setting is such that payments are not allowed (i.e., a public common good). According to [14, 119] mechanisms without monetary incentives have limited capabilities. A case by case study is the only hope for finding mechanisms with good properties and without payments, like in [88, 95]. FEDoR is a general mechanism that is truthful, fair and socially efficient, and without using monetary incentives. The inherent repeated nature of FEDoR , though, makes it applicable only to settings with infinite or unknown number of repetitions. If the number of repetitions is known to the agents, then they can devise a strategy that would maximize their utility and potentially lead to a non-truthful mechanism. Our intuition is that, if the nature of the problem allows it, mechanisms like FEDoR can be devised for other problems as well. Taking advantage of the repeated interaction between the agents, and applying the concept of linking mechanisms, should allow to derive other mechanisms without payments for multiple combinations of desirable properties (i.e., different fairness criteria, strategy proofness, social efficiency, etc). An additional advantage, as we have seen experimentally, is that a mechanism without payments like FEDoR was able, without incentivizing the workers through payments, to provide a way for the mechanism designer to select its utility and the utility of the participating players.

While FEDoR is a mechanism with several interesting features, there are some issues that are worth discussing. FEDoR assumes that the valuations of the players follow a uniform distribution. Although this might seem as a constraint, if the real bids do not follow such a distribution, they can be transformed by using the PIT, as proposed by Santos et al. [117]. Moreover we make the assumption that all players have i.i.d. valuations. As this is a first approach towards this line of mechanisms without monetary incentives we wanted to keep our model simple. In an extension of the mechanism, we plan to explore the correlated case where players distributions are correlated with those of other players [116].

An additional extension to the mechanism would be to allow players to provide their valuation for a particular good of the set and not for the whole set, as it is implemented now. This is yet another challenge, since players might have correlated preferences between certain goods in the same or different round.

In Section 6.5 we considered the case of FEDoR in an Internet-based computing system where the masters pay the same flat fee. There could be the case that masters with a higher budget would like to claim a larger number of workers over the consecutive rounds, something that FEDoR does not allow to happen due to the fairness property. An approach to solve this constrain of FEDoR would be to allow a master with a higher budget to participate with more than one identity. As a consequence, she will be increasing the amount of workers that she will receive, proportionally to the number of identities. Again though, allowing masters to have multiple identities one must assume that masters might have correlated preferences.

Non-Trivial Motivating Example In the field of computer science, FEDoR can be applied to solve the problem of efficiently allocating CPU cycles between processes. Another application could be the periodical allocation in a fair manner of bandwidth among nodes connected to the same point of access. Imagine the scenario where many nodes are connected to the same point of access. An application could be built to support the feature of nodes declaring how much they value receiving bandwidth from the point of access at a specific time interval. Then, nodes would be served from the point of access according to FEDoR . This would guarantee fairness and efficiency over time for all the nodes. More applications can be imagined, where FEDoR could allocate resources over the cloud. In this case it depends on the policy and the specific implementation on how FEDoR could apply. For example, a number of users with only periodic demands for large amount of resources could be sharing resources using FEDoR instead of resolving into costly provisioning of resources.

Another possible application is hospitals sharing valuable equipment. No allocation based on the money that each hospital is willing to give for renting the equipment can be made. Hence, FEDoR is a perfect candidate, without monetary payments can guarantee that all the hospitals sharing the valuable equipment will be allocated the same machine the same amount of time in expectation and most importantly in an efficient manner (when they most need it). Similar application could also be good candidates for FEDoR , as long of course no life threatening decisions are left to be taken by the mechanism.

We have mentioned before a number of intuitive examples where FEDoR could apply. What these examples have in common is the difficulty or inappropriateness of using payments to allocate the resources. This fact, in combination with the repeated nature of the setting makes FEDoR a perfect candidate in those scenarios. The good properties of the mechanism though (fairness, social efficiency and truthfulness) can make it a suitable candidate also to scenarios in which payments are essential for resource allocation, for example to sponsored search.

As the business model of search engines' (i.e., Google, Yahoo!, Bing, etc) has evolved over

time their main source of income are the ad links delivered to users each time they do a search. For each keyword search, there exists a number of available slots for advertisement in each page of results. Due to the fact that slots are limited while advertisers are numerous, and in combination with the search engines' goal for revenue maximization, auction mechanisms are used for slot allocation.

The setting of sponsored search fits the setting for which FEDoR was designed. A number of slots (goods in the general version) are allocated over r consecutive instances, having the same advertisers (in a simplified version of the problem) competing for them. Each slot can be seen as having a relative value. A simple analysis of the sponsored search auction [46] treats the click-through rate (CTR) of an advertisement as correlated with the slot positioning and hence this is where the relative value for each slot emerges.

Two famous auction mechanisms, as we mentioned before, are the VCG [16] and the GSP auction [46]. VCG is a powerful mechanism that is strategy proof and efficient (in every instance). In VCG the ith highest bidder gets the ith best good, but pays the *externality* that she imposes on the other bidders by winning that slot. A mechanism like that is difficult to be understood by a normal advertiser (it is also difficult to determine its valuation), making VCG not such a suitable candidate for sponsored search. On the other hand there is GSP, were the ith highest bidder gets the ith best good, but pays to the seller the bid of the $(i+1)$st highest bidder. This mechanism is neither strategy proof neither efficient but is easily understandable by the advertiser. In fact this mechanism was developed by Google and it is also used by Yahoo!. Adwords [1] of Google uses this mechanism but also takes into account a "quality score" [2]. Part of that quality score is the click through rate that a specific advertiser is estimated to obtain if she gets a specific slot. In this case, the allocation mechanism is basing its result on the product of the estimated CTR multiplied by the bid; and the payment of the bidder that wins a slot is obtained by multiplying her bid by the CTR. As mentioned in the previous paragraph we have modeled the CTR as a weight of the importance of each slot (like Edelman et al. [46]) and we assume that it is the same for all bidders.

The key business ingredient of the current sponsored search model is promoting high quality advertisement (measured through CTR estimation) and in parallel increase of revenue for the search engine. We argue that in addition to these features, fairness is a valid business model. Fairness, in the sense that no qualified advertiser would suffer publicity starvation (obtaining an ad slot few times or never). To this respect, FEDoR could be used to guarantee fairness. To deal with the payments that the search engine would require, a flat fee for participating in r rounds could be set.

The mechanism could run centrally on the search engine and advertisers would communicate their desire to appear to a certain keyword search. For r instances of this communication with the search engine the advertiser will pay a flat fee. FEDoR guarantees that in expectation the advertiser will appear the same number of times in each slot. It also guarantees truthfulness and social efficiency, meaning that if the advertisers provide their true valuations on the slots they will be receiving a slot when they most value it.

This business model could also apply to recommendation engines. Consider the example of Amazon's recommendation engine, that after a search for a good recommends to you several alternative goods. The way in which a decision is taken on which similar good should be presented to the customer can be based on the FEDoR mechanism, so that fairness among all similar goods will be guaranteed. A similar example is the one of showing adds to a newspaper reader related to the content of the news she read.

Comparing FEDoR to the celebrated VCG mechanism, it is clear that VCG has an advantage since its properties hold deterministically. On the other hand, FEDoR 's properties hold on expectation. This means that FEDoR can only be applied on a repeated setting where the agents remain the same. Nevertheless, simulations have shown us that besides fairness, applying FEDoR to a setting where the need of payments is inherent has another benefit. Due to the necessary flat participation fee scheme, the search engine (i.e., the seller) can set this fee in such a way to control her utility and the utility of the advertisers.

Chapter 7

Conclusions and Future Work

In this work, we proposed solutions for achieving reliability and fairness in online task computing environments. We modelled these environments though what we call Internet-based task computing system and through the master-worker paradigm we capture its two main components. To address the problem of reliability in Internet-based task computing we characterized the behavior of the workers by proposing two models: (1) the error probability model and (2) the rationality model. We supported our models' assumptions through the literature and through the experimental data we collected from posting tasks on Amazon Mechanical Turk (AMT). Thus, our modelling assumptions capture a realistic view of online task computing systems. This fact does not only provide an added value to our designed solutions, but also paves the way for more realistic assumptions in works exploring such online computing environments.

In Chapter 3, we present one of our two main modelling assumptions on the workers behavior. We consider the presence of altruistic and troll workers subject to an error probability that can alter their intended behavior. A generic model capturing the parameters of the master-worker paradigm is introduced that deviates from the usual conventions made in the literature, in two directions: the error probability that workers might have and the usual assumption that a task can only have one correct and one incorrect result, thus making our model more realistic. In our model we do not consider the possibility that workers might be unavailable at some stages in the execution, or that they may exhibit dynamic behavior by experiencing different error probability over time. We plan to address this issues in future work. We have assumed for simplicity of presentation that ϵ is the same for all workers, an assumption that also helped us gain a better understanding of our master-worker setting. In the future we plan to remove this assumption and consider that each worker has its own error probability that can even change over time. Thus, we plan to adapt our algorithm to deal with this dynamic setting by using an appropriate reputation technique to keep track and update the workers error probability.

The mechanisms designed in Chapter 4 take a step forward towards being applied to volunteer computing or crowdsourcing systems, since they do not make any assumption on the distribution of worker types in the system. We take advantage of the repeated interaction of the workers with

the master and we apply reinforcement learning techniques to capture the evolution of Internet-based master-worker computations. Although the assumption that a task has a unique solution might seem limiting, indeed many tasks in volunteer computing and crowdsourcing systems have unique solutions. Moreover, in crowdsourcing systems the task can often be designed in such a way that only one correct solution exist. We model the workers behavior assuming the presence of malicious, atruistic and rational workers and we show that under necessary and sufficient conditions, the master reaches a state after which the correct task result is obtained at each round, with minimal cost, that is, a state of eventual correctness. In addition, we show analytically that in the presence of only rational workers such state can be reached ''quickly''. Moreover, we present a malicious-tolerant generic mechanism that uses reputation. We consider four reputation types, and give provable guarantees that only reputation Exponential (introduced in this work) provides eventual correctness. Finally, we design reputation mechanism dealing with workers being unresponsive.

Our simulations considering the presence of only rational workers suggest that having a positive reinforcement learning (i.e., $WP_{\mathcal{C}} = 0$) the master can reach fast convergence, while applying punishments (i.e., $WP_{\mathcal{C}} \in \{1, 2\}$) provides even faster convergence. In fact, we may conclude that applying only punishment is enough to have fast convergence in the presence of rational workers. Also, our simulations demonstrated that it is possible when the master begins with an aggressive auditing strategy, to have a smaller cost and convergence time compared to the case of using a less aggressive auditing policy. Simulating the system in the presence of malicious, altruistic and rational workers showed that having all rational workers covered is enough to reach eventual correctness in all four reputation types. In the case of covering only one altruistic or rational worker, simulations showed that only reputation Exponential can achieve eventual correctness. We show that reputation Exponential has more potential in commercial platforms where high reliability together with low auditing cost, rewarding few workers and fast convergence are required. We believe this advances the development of reliable commercial Internet-based task computing systems. In particular, our simulations reveal interesting tradeoffs between reputation types and parameters and show that our mechanism is a generic one that can be adjusted to various settings. Additionally, we have seen that when full availability is considered, reputation Boinc is not a desirable reputation type if the pool of workers is large. This is an interesting result since reputation Boinc is an adaptation of the reputation used in volunteer computing systems dealing with large pools of workers. Finally, it is interesting to observe that, in the partial availability case with only rational workers, our mechanism did not reach eventual correctness when the system converged, but a few rounds later. This means that, although the rational workers are partially available, the mechanism is able to reinforce them to behave honestly eventually.

The analysis of the system in Chapter 4 is done assuming that workers have an implicit form of collusion, i.e., we assume that all misbehaving workers reply with the same answer and all workers behaving correctly give the same answer. Following [53], we are now studying stronger models, in which workers collude in deciding when to cheat and when to be honest. In a follow-

up work we plan to investigate what happens if workers are connected to each other, forming a network (i.e, a social network through which they can communicate) or if malicious workers develop a more intelligent strategy against the system. Also the degree of trust among the players has to be considered and modeled in this scenario.

In our experimental evaluation we were able to classify the workers behavior and draw conclussions on the correlation among time invested on the Human Intelligence Task (HIT) and correctness of the answer to the HIT. We have also been able to identify several of the behaviors we postulated in our models, thus showing that they are realistic enough to understand already existing commercial crowdsourcing platforms such as AMT. We have looked at tasks that had a unique correct solution. In the future, we would like to evaluate the workers behavior where a task result is evaluated based on a quality measure rather than having a yes or no response. To this respect, new mechanisms need to be designed incentivizing the workers to provide high quality results. Moreover, we wish to perform an intensive series of experiments considering different types of tasks [47] and try to systematically classify the participating workers into a number of different classes based on their error rate. It has been shown [129] that when such information is available it can be exploited during the task assignment process and significantly improve the system performance.

Finally, we have designed Fair and Efficient Distribution of Resources (FEDoR), a mechanism for achieving fair, socially efficient and truthful allocation of the workers among the masters in expectation over multiple rounds, without the use of monetary incentives. The inherent repeated nature of FEDoR , makes it applicable to other settings with infinite or unknown number of repetitions. If the number of repetitions is known to the strategic players, then they can devise a strategy that would maximize their utility and potentially lead to a non-truthful mechanism. Our intuition is that, if the nature of the problem allows it, mechanisms like FEDoR can be devised for other problems as well. Taking advantage of the repeated interaction between the players, and applying the concept of linking mechanisms, should allow to derive other mechanisms without payments for multiple combinations of desirable properties (i.e., different fairness criteria, strategy proofness, social efficiency, etc). An additional advantage, as we have seen experimentally, is that a mechanism without payments like FEDoR was able, without incentivizing the workers through payments, to provide a way for the mechanism designer to select its utility and the utility of the participating masters. According to [14, 119] mechanisms without monetary incentives have limited capabilities. A case by case study is the only hope for finding mechanisms with good properties and without payments, like in [88, 95].

While FEDoR is a mechanism with several interesting features, there are some issues that are worth discussing. FEDoR assumes that the valuations of the players follow a uniform distribution. Although this might look like as a constraint, if the real bids do not follow such a distribution, they can be transformed by using the Probability Integral Transformation (PIT) transformation, as proposed by Santos et al. [117]. Moreover, we make the assumption that all players have i.i.d. valuations. As this is a first approach towards this line of mechanisms without monetary incentives

we wanted to keep our model simple. In an extension of the mechanism, we plan to explore the correlated case where players distributions are correlated with those of other players [116].

In our particular master-worker scenario we have considered that all players (a.k.a. masters) pay the same flat fee. There could be the case that masters with a higher budget would like to claim a larger number of workers over the consecutive rounds, something that FEDoR does not allow to happen due to the fairness property. An approach to solve this constraint of FEDoR would be to allow a master with a higher budget to participate with more than one identity. As a consequence, she will be increasing the amount of task allocations to workers that she will receive, proportionally to the number of identities. However, allowing masters to have multiple identities one must assume that that masters might have correlated preferences.

References

[1] Google."google adwords". www.adwords.google.com/, 2017.

[2] Google."how are ads ranked". http://bit.ly/2r5FQr8, 2017.

[3] Copyright IBM Corporation 2017. World Community Grid, 2017. `https://secure.worldcommunitygrid.org/`.

[4] Ittai Abraham, Danny Dolev, Rica Gonen, and Joe Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of Distributed Computing*, pages 53–62. ACM, 2006.

[5] Amitanand S Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. Bar fault tolerance for cooperative services. In *ACM SIGOPS Operating Systems Review*, volume 39, pages 45–58. ACM, 2005.

[6] Bruce Allen. The Einstein@home project, 2016. `http://einstein.phys.uwm.edu`.

[7] Noga Alon, Yuval Emek, Michal Feldman, and Moshe Tennenholtz. Bayesian ignorance. *Theor. Comput. Sci.*, 452:1–11, 2012.

[8] David P Anderson. Boinc: A system for public-resource computing and storage. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 4–10. IEEE, 2004.

[9] David P Anderson. Volunteer computing: the ultimate cloud. *ACM Crossroads*, 16(3):7–10, 2010.

[10] David P Anderson. BOINC adaptive replication legacy, 2014. `https://boinc.berkeley.edu/trac/wiki/AdaptiveReplication?version=8`.

[11] David P Anderson. BOINC adaptive replication, 2017. `http://boinc.berkeley.edu/trac/wiki/AdaptiveReplication`.

[12] David P Anderson and Kevin Reed. Celebrating diversity in volunteer computing. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, pages 1–8. IEEE, 2009.

[13] John E Angus. The probability integral transform and related results. *SIAM review*, 36(4):652–654, 1994.

[14] Kenneth J Arrow. *Social choice and individual values*, volume 12. Yale university press, 2012.

[15] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[16] Lawrence M Ausubel and Paul Milgrom. The lovely but lonely vickrey auction. *Combinatorial auctions*, pages 17–40, 2006.

[17] Moshe Babaioff, Michal Feldman, and Noam Nisan. Combinatorial agency. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 18–28. ACM, 2006.

[18] Moshe Babaioff, Michal Feldman, and Noam Nisan. Mixed strategies in combinatorial agency. In *International Workshop on Internet and Network Economics*, pages 353–364. Springer, 2006.

[19] Moshe Babaioff, Michal Feldman, and Noam Nisan. Free-riding and free-labor in combinatorial agency. In *International Symposium on Algorithmic Game Theory*, pages 109–121. Springer, 2009.

[20] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better-how to make bitcoin a better currency. In *Financial Cryptography and Data Security*, pages 399–414. Springer, 2012.

[21] Jonathan Bendor, Dilip Mookherjee, and Debraj Ray. Aspiration-based reinforcement learning in repeated interaction games: An overview. *International Game Theory Review*, 3(02n03):159–174, 2001.

[22] Jonathan Bendor, Dilip Mookherjee, and Debraj Ray. Reinforcement learning in repeated interaction games. *Advances in Theoretical Economics*, 1(1), 2001.

[23] BitcoinMining.com. Bitcoin mining, 2017. `http://www.bitcoinmining.com`.

[24] Douglas M Blough and Gregory F Sullivan. A comparison of voting strategies for fault-tolerant distributed systems. In *Reliable Distributed Systems, 1990. Proceedings., Ninth Symposium on*, pages 136–145. IEEE, 1990.

[25] John Bohannon. Mechanical turk upends social sciences. *Science*, 352(6291):1263–1264, 2016.

[26] Tilman Börgers and Rajiv Sarin. Learning through reinforcement and replicator dynamics. *Journal of Economic Theory*, 77(1):1–14, 1997.

[27] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. Amazon's mechanical turk a new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011.

[28] Robert R Bush and Frederick Mosteller. Stochastic models for learning. 1955.

[29] Colin Camerer. *Behavioral game theory: Experiments in strategic interaction*. Princeton University Press, 2003.

[30] Steve Chien and Alistair Sinclair. Convergence to approximate nash equilibria in congestion games. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 169–178. Society for Industrial and Applied Mathematics, 2007.

[31] Evgenia Christoforou, Antonio Fernández Anta, Chryssis Georgiou, and Miguel A. Mosteiro. Algorithmic mechanisms for reliable master-worker internet-based computing. *IEEE Trans. Computers*, 63(1):179–195, 2014.

[32] Evgenia Christoforou, Antonio Fernández Anta, Chryssis Georgiou, and Miguel A. Mosteiro. Internet computing: Using reputation to select workers from a pool. In *Networked Systems - 4th International Conference, NETYS 2016, Marrakech, Morocco, May 18-20, 2016, Revised Selected Papers*, pages 137–153, 2016.

[33] Evgenia Christoforou, Antonio Fernández Anta, Chryssis Georgiou, Miguel A. Mosteiro, and Ángel Sánchez. Achieving reliability in master-worker computing via evolutionary dynamics. In *Euro-Par 2012 Parallel Processing - 18th International Conference, Euro-Par 2012, Rhodes Island, Greece, August 27-31, 2012. Proceedings*, pages 451–463, 2012.

[34] Evgenia Christoforou, Antonio Fernández Anta, Chryssis Georgiou, Miguel A. Mosteiro, and Ángel Sánchez. Brief announcement: achieving reliability in master-worker computing via evolutionary dynamics. In *ACM Symposium on Principles of Distributed Computing, PODC '12, Funchal, Madeira, Portugal, July 16-18, 2012*, pages 225–226, 2012.

[35] Evgenia Christoforou, Antonio Fernández Anta, Chryssis Georgiou, Miguel A. Mosteiro, and Ángel Sánchez. Applying the dynamics of evolution to achieve reliability in master-worker computing. *Concurrency and Computation: Practice and Experience*, 25(17):2363–2380, 2013.

[36] Evgenia Christoforou, Antonio Fernández Anta, Chryssis Georgiou, Miguel A Mosteiro, and Angel Sánchez. Crowd computing as a cooperation problem: an evolutionary approach. *Journal of Statistical Physics*, 151(3-4):654–672, 2013.

[37] Evgenia Christoforou, Antonio Fernández Anta, Chryssis Georgiou, Miguel A. Mosteiro, and Ángel Sánchez. Reputation-based mechanisms for evolutionary master-worker computing. In *Principles of Distributed Systems - 17th International Conference, OPODIS 2013, Nice, France, December 16-18, 2013. Proceedings*, pages 98–113, 2013.

[38] Evgenia Christoforou, Antonio Fernández Anta, Kishori M. Konwar, and Nicolas C. Nicolaou. Evaluating reliability techniques in the master-worker paradigm. In *15th IEEE International Symposium on Network Computing and Applications, NCA 2016, Cambridge, Boston, MA, USA, October 31 - November 2, 2016*, pages 183–190, 2016.

[39] Evgenia Christoforou, Antonio Fernández Anta, and Agustín Santos. A mechanism for fair distribution of resources with application to sponsored search. *CoRR*, abs/1502.03337, 2015.

[40] Evgenia Christoforou, Antonio Fernández Anta, and Agustín Santos. A mechanism for fair distribution of resources without payments. *PLOS ONE*, 11(5):1–20, 05 2016.

[41] Daniel Clery. Galaxy zoo volunteers share pain and glory of research. *Science*, 333(6039):173–175, 2011.

[42] Vincent Conitzer and Tuomas Sandholm. Incremental mechanism design. In *IJCAI*, pages 1251–1256, 2007.

[43] P. Dagum, R.M. Karp, M. Luby, and S. Ross. An optimal algorithm for monte carlo estimation. In *Proceedings of the Foundations of Computer Science*, pages 142–149, 1995.

[44] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. Mechanical cheat: Spamming schemes and adversarial techniques on crowdsourcing platforms. In *CrowdSearch*, pages 26–30, 2012.

[45] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

[46] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. Technical report, National Bureau of Economic Research, 2005.

[47] Carsten Eickhoff and Arjen P de Vries. Increasing cheat robustness of crowdsourcing tasks. *Information retrieval*, 16(2):121–137, 2013.

[48] Raphael Eidenbenz and Stefan Schmid. Combinatorial agency with audits. In *Game Theory for Networks, 2009. GameNets' 09. International Conference on*, pages 374–383. IEEE, 2009.

[49] Kfir Eliaz. Fault tolerant implementation. *The Review of Economic Studies*, 69(3):589–610, 2002.

[50] Trilce Estrada, Michela Taufer, and David P Anderson. Performance prediction and analysis of boinc projects: An empirical study with emboinc. *Journal of Grid Computing*, 7(4):537–554, 2009.

[51] Antonio Fernández Anta, Chryssis Georgiou, Luis López, and Agustin Santos. Reliable internet-based master-worker computing in the presence of malicious workers. *Parallel Processing Letters*, 22(01), 2012.

[52] Antonio Fernández Anta, Chryssis Georgiou, and Miguel A Mosteiro. Designing mechanisms for reliable internet-based computing. In *Network Computing and Applications, 2008. NCA'08. Seventh IEEE International Symposium on*, pages 315–324. IEEE, 2008.

[53] Antonio Fernández Anta, Chryssis Georgiou, Miguel A Mosteiro, and Daniel Pareja. Algorithmic mechanisms for reliable crowdsourcing computation under collusion. *PloS one*, 10(3), 2015.

[54] Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1631–1640. ACM, 2015.

[55] Martin Gairing. Malicious bayesian congestion games. In *Approximation and Online Algorithms, 6th International Workshop, WAOA 2008, Karlsruhe, Germany, September 18-19, 2008. Revised Papers*, pages 119–132, 2008.

[56] GalaxyZoo. Galaxy Zoo Website, 2017. `https://www.galaxyzoo.org/`.

[57] Herbert Gintis. *Game theory evolving: A problem-centered introduction to modeling strategic behavior*. Princeton University Press, 2000.

[58] Philippe Golle and Ilya Mironov. Uncheatable distributed computations. In *Topics in Cryptology?CT-RSA 2001*, pages 425–440. Springer, 2001.

[59] Mingyu Guo and Vincent Conitzer. Strategy-proof allocation of multiple items between two agents without payments or priors. In *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3*, pages 881–888, 2010.

[60] Mingyu Guo, Vincent Conitzer, and Daniel M. Reeves. Competitive repeated allocation without payments. In *Internet and Network Economics, 5th International Workshop, WINE 2009, Rome, Italy, December 14-18, 2009. Proceedings*, pages 244–255, 2009.

[61] John C Harsanyi, Reinhard Selten, et al. A general theory of equilibrium selection in games. *MIT Press Books*, 1, 1988.

[62] Jason D Hartline and Tim Roughgarden. Optimal mechanism design and money burning. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 75–84. ACM, 2008.

[63] Eric Martin Heien, David P Anderson, and Kenichi Hagihara. Computing low latency batches with unreliable workers in volunteer computing environments. *Journal of Grid Computing*, 7(4):501–518, 2009.

[64] Joseph Henrich and Richard McElreath. The evolution of cultural evolution. *Evolutionary Anthropology: Issues, News, and Reviews*, 12(3):123–135, 2003.

[65] Martin Hoefer and Alexander Skopalik. Altruism in atomic congestion games. *ACM Trans. Economics and Comput.*, 1(4):21, 2013.

[66] John Joseph Horton and Lydia B Chilton. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 209–218. ACM, 2010.

[67] Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[68] Paul Hyman. Software aims to ensure fairness in crowdsourcing projects. *Commun. ACM*, 56(8):19–21, 2013.

[69] Panagiotis G Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students*, 17(2):16–21, 2010.

[70] Luis R Izquierdo and Segismundo S Izquierdo. Dynamics of the bush-mosteller learning algorithm in 2x2 games. 2008.

[71] Matthew O Jackson and Hugo F Sonnenschein. Overcoming incentive constraints by linking decisions1. *Econometrica*, 75(1):241–257, 2007.

[72] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.

[73] David R Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*, pages 1953–1961, 2011.

[74] Nicolas Kaufmann, Thimo Schulze, and Daniel Veit. More than fun and money. worker motivation in crowdsourcing-a study on mechanical turk. In *AMCIS*, volume 11, pages 1–11, 2011.

[75] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–456. ACM, 2008.

[76] Laure Kloetzer, Daniel Schneider, Charlene Jennett, Ioanna Iacovides, Alexandra Eveleigh, Anna Cox, and Margaret Gold. Learning by volunteer computing, thinking and gaming: What and how are volunteers learning by participating in virtual citizen science? *Changing Configurations of Adult Education in Transitional Times*, page 73, 2014.

[77] Derrick Kondo, Filipe Araujo, Paul Malecot, Patricio Domingues, Luis Moura Silva, Gilles Fedak, and Franck Cappello. Characterizing result errors in internet desktop grids. In *Euro-Par 2007 Parallel Processing*, pages 361–371. Springer, 2007.

[78] Kishori M. Konwar, Sanguthevar Rajasekaran, and Alexander A. Shvartsman. Robust network supercomputing with unreliable workers. *J. Parallel Distrib. Comput.*, 75:81–92, 2015.

[79] Eric Korpela, Dan Werthimer, David P Anderson, Jeff Cobb, and Matt Lebofsky. Seti@ home: massively distributed computing for seti. *Computing in science & engineering*, 3(1):78–83, 2001.

[80] Vijay Krishna. *Auction theory*. Academic press, 2009.

[81] Michael Kuhn, Stefan Schmid, and Roger Wattenhofer. Distributed asymmetric verification in computational grids. In *22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, Miami, Florida USA, April 14-18, 2008*, pages 1–10, 2008.

[82] Akhil Kumar and Kavindra Malik. Voting mechanisms in distributed systems. *Reliability, IEEE Transactions on*, 40(5):593–600, 1991.

[83] Petr Kuznetsov and Stefan Schmid. Towards network games with social preferences. In *Structural Information and Communication Complexity, 17th International Colloquium, SIROCCO 2010, Sirince, Turkey, June 7-11, 2010. Proceedings*, pages 14–28, 2010.

[84] Juan Luis Jiménez Laredo, Pascal Bouvry, D Lombraña González, F Fernández De Vega, Maribel García Arenas, JJ Merelo, and Carlos M Fernandes. Designing robust volunteer-based evolutionary algorithms. *Genetic Programming and Evolvable Machines*, 15(3):221–244, 2014.

[85] Jean-François Laslier, Richard Topol, and Bernard Walliser. A behavioral learning process in games. *Games and Economic Behavior*, 37(2):340–366, 2001.

[86] Jerald F Lawless. *Statistical models and methods for lifetime data*, volume 362. John Wiley & Sons, 2011.

[87] Daniel Lázaro, Derrick Kondo, and Joan Manuel Marquès. Long-term availability prediction for groups of volunteer resources. *Journal of Parallel and Distributed Computing*, 72(2):281–296, 2012.

[88] Hagay Levin, Michael Schapira, and Aviv Zohar. Interdomain routing and games. *SIAM Journal on Computing*, 40(6):1892–1912, 2011.

[89] Harry C Li, Allen Clement, Mirco Marchetti, Manos Kapritsos, Luke Robison, Lorenzo Alvisi, and Mike Dahlin. Flightpath: Obedience vs. choice in cooperative services. In *OSDI*, volume 8, pages 355–368, 2008.

[90] Harry C Li, Allen Clement, Edmund L Wong, Jeff Napper, Indrajit Roy, Lorenzo Alvisi, and Michael Dahlin. Bar gossip. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 191–204. USENIX Association, 2006.

[91] Thomas Locher, Patrick Moor, Stefan Schmid, and Roger Wattenhofer. Free riding in bittorrent is cheap. In *Proc. Workshop on Hot Topics in Networks (HotNets)*, pages 85–90. Citeseer, 2006.

[92] Andreu Mas-Colell, Michael Dennis Whinston, Jerry R Green, et al. *Microeconomic theory*, volume 1. Oxford university press New York, 1995.

[93] Winter Mason and Siddharth Suri. Conducting behavioral research on amazon's mechanical turk. *Behavior research methods*, 44(1):1–23, 2012.

[94] Winter Mason and Duncan J Watts. Financial incentives and the performance of crowds. *ACM SigKDD Explorations Newsletter*, 11(2):100–108, 2010.

[95] Thomas Moscibroda and Stefan Schmid. On mechanism design without payments for throughput maximization. In *INFOCOM 2009. 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 19-25 April 2009, Rio de Janeiro, Brazil*, pages 972–980, 2009.

[96] Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. When selfish meets evil: byzantine players in a virus inoculation game. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, pages 35–44, 2006.

[97] Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981.

[98] John F Nash et al. Equilibrium points in n-person games. *Proc. Nat. Acad. Sci. USA*, 36(1):48–49, 1950.

[99] Noam Nisan and Amir Ronen. Algorithmic mechanism design. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 129–140. ACM, 1999.

[100] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007.

[101] Martin A Nowak. *Evolutionary dynamics*. Harvard University Press, 2006.

[102] Amazon.com Inc. or its Affiliates. Amazon's Mechanical Turk, 2005-2017. `https://www.mturk.com`.

[103] Martin J Osborne. *An introduction to game theory*, volume 3. Oxford university press New York, 2004.

[104] Gabriele Paolacci and Jesse Chandler. Inside the turk: Understanding mechanical turk as a participant pool. *Current Directions in Psychological Science*, 23(3):184–188, 2014.

[105] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. 2010.

[106] Michel Paquette and Andrzej Pelc. Optimal decision strategies in byzantine environments. *Journal of Parallel and Distributed Computing*, 66(3):419–427, 2006.

[107] Eyal Peer, Joachim Vosgerau, and Alessandro Acquisti. Reputation as a sufficient condition for data quality on amazon mechanical turk. *Behavior research methods*, 46(4):1023–1031, 2014.

[108] Steve Phelps, Peter McBurney, and Simon Parsons. Evolutionary mechanism design: a review. *Autonomous Agents and Multi-Agent Systems*, 21(2):237–264, 2010.

[109] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in bittorrent. In *Proc. of NSDI*, volume 7, 2007.

[110] Ariel D. Procaccia and Moshe Tennenholtz. Approximate mechanism design without money. *ACM Trans. Economics and Comput.*, 1(4):18, 2013.

[111] Rameez Rahman, Michel Meulpolder, David Hales, Johan Pouwelse, Dick Epema, and Henk Sips. Improving efficiency and fairness in p2p systems with effort-based incentives. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5. IEEE, 2010.

[112] David Rose and Thomas R Willemain. The principal-agent problem with evolutionary learning. *Computational & Mathematical Organization Theory*, 2(2):139–162, 1996.

[113] Robert W Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[114] Steven V Rouse. A reliability analysis of mechanical turk data. *Computers in Human Behavior*, 43:304–307, 2015.

[115] Larry Samuelson. *Evolutionary games and equilibrium selection*, volume 1. Mit Press, 1998.

[116] Agustín Santos, Antonio Fernández Anta, José A. Cuesta, and Luis López Fernández. Fair linking mechanisms for resource allocation with correlated player types. In *Networked Systems - Second International Conference, NETYS 2014, Marrakech, Morocco, May 15-17, 2014. Revised Selected Papers*, pages 70–83, 2014.

[117] Agustín Santos, Antonio Fernández Anta, and Luis López Fernández. Quid Pro Quo: a mechanism for fair collaboration in networked systems. *PloS one*, 8(9):e66575, 2013.

[118] Luis FG Sarmenta. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems*, 18(4):561–572, 2002.

[119] Mark Allen Satterthwaite. Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of economic theory*, 10(2):187–217, 1975.

[120] J. Schummer and R. V. Vohra. Mechanism design without money. In E. Tardos N. Nisan, T. Roughgarden and V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, Cambridge, 2007.

[121] Sebastien. WUProp@home project, 2017. `http://wuprop.boinc-af.org/`.

[122] SETI@home. SETI@home Poll Results, 2000. `http://boinc.berkeley.edu/slides/xerox/polls.html`.

[123] Aaron D Shaw, John J Horton, and Daniel L Chen. Designing incentives for inexpert human raters. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pages 275–284. ACM, 2011.

[124] Jeffrey Shneidman and David C Parkes. Rationality and self-interest in peer to peer networks. In *Peer-to-Peer Systems II*, pages 139–148. Springer, 2003.

[125] J Maynard Smith and GR Price. lhe logic of animal conflict. *Nature*, 246:15, 1973.

[126] John Maynard Smith. *Evolution and the Theory of Games*. Cambridge university press, 1982.

[127] Jason Sonnek, Abhishek Chandra, and Jon B Weissman. Adaptive reputation-based scheduling on unreliable distributed infrastructures. *Parallel and Distributed Systems, IEEE Transactions on*, 18(11):1551–1564, 2007.

[128] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, 2010.

[129] Alberto Tarable, Alessandro Nordio, Emilio Leonardi, and Marco Ajmone Marsan. The importance of worker reputation information in microtask-based crowd work systems. *IEEE Trans. Parallel Distrib. Syst.*, 28(2):558–571, 2017.

[130] Michela Taufer, David P Anderson, Pietro Cicotti, and Charles L Brooks III. Homogeneous redundancy: a technique to ensure integrity of molecular simulation results using public computing. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 119a–119a. IEEE, 2005.

[131] Luis Von Ahn. Games with a purpose. *Computer*, 39(6):92–94, 2006.

[132] Jörgen W Weibull. *Evolutionary game theory*. MIT press, 1997.

[133] Matthew Yurkewych, Brian N Levine, and Arnold L Rosenberg. On the cost-ineffectiveness of redundancy in commercial P2P computing. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 280–288. ACM, 2005.

[134] Yu Zhang and Mihaela van der Schaar. Reputation-based incentive protocols in crowd-sourcing applications. In *INFOCOM, 2012 Proceedings IEEE*, pages 2140–2148. IEEE, 2012.

[135] Shanyu Zhao, Virginia Lo, and C Gauthier Dickey. Result verification and trust-based scheduling in peer-to-peer grids. In *Peer-to-Peer Computing, 2005. P2P 2005. Fifth IEEE International Conference on*, pages 31–38. IEEE, 2005.

Appendix

Publications

The work presented in this doctoral thesis has appeared in three journal articles [35, 36, 40], four conference papers [32, 33, 37, 38], one poster [39], one brief announcement [34] and one workshop paper [39]. Additionally, one more journal article is under review. In this section we briefly elaborate on these publications and how they form part of the doctoral thesis.

We have published our work related to reinforcing the behavior of the rational workers and achieving a reliable computational platform in three conferences. Each of the three conference publications (presented below) was extending on the original ideal of taking advantage of the repeated interaction among the master and the workers to achieve a reliable computational platform. In the first conference work we presented the case where only rational workers where present, while in the two subsequent conference papers we considered the presence of malicious and rational workers, as well as the case of unresponsive workers.

- **E. Christoforou**, A. Fernández Anta, Ch. Georgiou, M. A. Mosteiro and Angel Sánchez, Achieving Reliability in Master-Worker Computing via Evolutionary Dynamics, in the Proceedings of the International European Conference on Parallel and Distributed Computing (EURO-PAR), Rhodes Island, Greece, 2012.

- **E. Christoforou**, A. Fernández Anta, Ch. Georgiou, M. A. Mosteiro and Angel Sánchez, Reputation-based Mechanisms for Evolutionary Master-Worker Computing, in the Proceedings of the 17th International Conference On Principles Of Distributed Systems (OPODIS), Nice, France, 2013.

- **E. Christoforou**, A. Fernández Anta, Ch. Georgiou and M. A. Mosteiro, Internet Computing: Using Reputation to Select Workers from a Pool, in the Proceedings of the 4th International Conference on Networked Systems (Netys), Marrakech, Morocco, 2016.

The work presented on the first conference paper was extended and appeared in two journal articles, presented below. The journal version of the second conference paper considering the presence of malicious, rational and altruistic workers is currently under review on ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS).

- **E. Christoforou**, A. Fernández Anta, Ch. Georgiou, M. A. Mosteiro and Angel Sánchez, Applying the dynamics of evolution to achieve reliability in master-worker computing, Journal of Concurrency and Computation: Practice and Experience, 2013 (invited extended version of the EURO-PAR 2012 article).

- **E. Christoforou**, A. Fernández Anta, Ch. Georgiou, M. A. Mosteiro and Angel Sánchez, Crowd Computing as a Cooperation Problem: An Evolutionary Approach, Journal of Statistical Physics, 2013.

Additionally, the preliminary results of the original work achieving a reliable computational platform in the presence of rational workers were presented in a brief announcement.

- **E. Christoforou**, A. Fernández Anta, Ch. Georgiou, M. A. Mosteiro and Angel Sánchez, Brief Announcement: Achieving Reliability in Master-Worker Computing via Evolutionary Dynamics, in the Proceedings of the 31st Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), Madeira, Portugal, 2012.

Moreover, we have published our fundamental results evaluating the reliability techniques in online task computing environments in the presence of unreliability and tasks with multiple correct and multiple incorrect solutions.

- **E. Christoforou**, A. Fernández Anta, K. M. Konwar and N. Nicolaou, Evaluating reliability techniques in the master-worker paradigm, in the Proceedings of the IEEE 15th International Symposium on Network Computing and Applications (IEEE NCA16), Cambridge, MA, 2016.

The second part of this work related to fair and efficient allocation of workers to master entities under the master-worker paradigm appeared in a journal version where our designed mechanism was presented under a general framework. In this journal version, we presented the applicability of our designed mechanism in a vast number of resource allocation problems in the presence of strategic agents and without the use of monetary incentives.

- **E. Christoforou**, A. Fernández Anta and A. Santos, A Mechanism for Fair Distribution of Resources without Payments, PLOS ONE, 2016.

Finally, preliminary results of the above mentioned work where disseminated as a poster and a workshop paper.

- **E. Christoforou**, A. Fernández Anta, A. Santos, A Mechanism for Fair Distribution of Resources with Application to Sponsored Search, Poster presented in the 10th Conference on Web and Internet Economics (WINE 2014), Beijing, China, 2014.

- **E. Christoforou**, A. Fernández Anta and A. Santos. A Mechanism for Fair Distribution of Resources with Application to Sponsored Search, in the Proceedings of the XXIII Jornadas de Concurrencia y Sistemas Distribuidos, Malaga, Spain, 2015.