

# TurkServer: Enabling Synchronous and Longitudinal Online Experiments

# TurkServer: Enabling Synchronous and Longitudinal Online Experiments

**Andrew Mao**
Harvard University
mao@seas.harvard.edu

**Yiling Chen**
Harvard University
yiling@eecs.harvard.edu

**Krzysztof Z. Gajos**
Harvard University
kgajos@eecs.harvard.edu

**David C. Parkes**
Harvard University
parkes@eecs.harvard.edu

**Ariel D. Procaccia**
Carnegie Mellon University
arielpro@cs.cmu.edu

**Haoqi Zhang**
Harvard University
hq@eecs.harvard.edu

**Abstract**

With the proliferation of online labor markets and other social computing platforms, online experiments have become a low-cost and scalable way to empirically test hypotheses and mechanisms in both human computation and social science. Yet, despite the potential in designing more powerful and expressive online experiments using multiple subjects, researchers still face many technical and logistical difficulties. We see *synchronous* and *longitudinal* experiments involving real-time interaction between participants as a dual-use paradigm for both human computation and social science, and present *TurkServer*, a platform that facilitates these types of experiments on Amazon Mechanical Turk. Our work has the potential to make more fruitful online experiments accessible to researchers in many different fields.

## Introduction

Online labor markets—Amazon Mechanical Turk (MTurk) being the paradigmatic example—have emerged as popular crowdsourcing platforms where requesters provide tasks for workers to complete at various compensations and effort levels. MTurk in particular is a market for microtasks, simple tasks with compensation ranging from a few cents to a few dollars per task. With easy access to a large and constant pool of workers at a low cost and quick responsiveness, MTurk has become not only an effective platform for crowdsourcing and human computation, but also a testbed for experimentalists in economics, sociology, psychology, and computer science, among other disciplines. In this paper, we present a software platform, *TurkServer*, that automates and streamlines complex experiments involving numerous, possibly real-time interactions between workers on MTurk.

Such an experimental platform is important and desirable for at least two reasons. First, the design of human computation systems will benefit from extensive experimental studies on the often complex interactions between individuals. Human computation has outgrown its original definition of rather simple tasks exemplified by *games with a purpose* (von Ahn and Dabbish 2008), designed so that people can contribute useful computations as a by-product of playing an enjoyable game. Many problems in human computation now fall into the paradigm of simultaneous or sequential interaction (Bernstein et al. 2011; Zhang et al. 2012) or iterative processes (Little et al. 2010a). Shahaf and Horvitz (2010) proposed task markets with even more heterogeneity, using agents with both human and machine intelligence and appropriate assignment of tasks. As the field moves forward and deals with more nuanced problems that require higher rates of participation and even real-time collaboration, we inevitably face the question of how to design effective human computation systems. An experimental approach is the essential first step to developing a theory of design for human computation.

Second, the experimental platform can benefit research in the social sciences, where MTurk is now recognized as a legitimate platform for conducting behavioral experiments. Many classic social experiments have been reproduced in the online setting (Horton, Rand, and Zeckhauser 2011; Amir, Rand, and Gal 2012; Rand 2012), suggesting the validity of experimentation on MTurk. In addition, MTurk makes a more culturally diverse subject pool available than typical university labs, and the ability to identify distinct workers (though anonymously) makes longitudinal studies possible (Paolacci, Chandler, and Ipeirotis 2010). However, undertaking real-time experiments online such as trading in experimental asset markets (Plott and Sunder 1982) or performing larger-scale field experiments for testing or validating theoretical results still involves significant implementation and effort.

This growing trend toward more interaction and participation in experimentation is at odds with the current paradigm of use on MTurk, where the vast majority of both experimentation and computation tasks involve very simple and typically one-shot interactions with workers, such as image labeling, translation, audio transcription, text recognition, and answering survey questions. We see an opportunity for researchers in both human computation and social science to make use of experimental methodologies on MTurk that support more complex interactions between participants, and focus on two particular types in this paper. *Synchronous experiments* are analogous to traditional laboratory settings, where experimenters can schedule a session consisting of many different rounds or treatments in which

subjects participate. *Longitudinal experiments* allow many subjects to interact with a central system or mechanism over time and hence indirectly with each other. *TurkServer* is a software platform that sigificantly reduces the logistical and technical obstacles to conducting these online experiments, making the benefits of scale, population diversity, low cost, and fast turnaround more readily available.

### Synchronous Experiments

Online synchronous experiments are most comparable to laboratory experiments supported by software such as z-Tree (Fishbacher 2006). A laboratory experiment session typically consists of a sequence of short rounds that require frequent or real-time interaction between a set of subjects seated in the same lab. In the online setting, however, the physical capacity of the lab no longer constrains the number of subjects in an experiment, nor must the sessions take place with a fixed set of subjects that interact with each other in progressive rounds. It becomes possible to run experiments of significantly larger scale, and experiment rounds can take place with subjects grouped as soon as they are available, as long as this doesn't interfere with assumptions on how data is collected. This is well suited for the typical behavioral research model of understanding how humans interact in different environments or mechanisms, where subjects are briefed before each session and their behavior is not expected to vary greatly over repeated rounds.

### Longitudinal Experiments

A more unique direction that we envision is to conduct online longitudinal experiments, which enable the observation and analysis of long-term effects of an intervention. By tracking a large group of users that interact asynchronously over time, we can conduct experiments that characterize different behaviors in a setting close to the real world and empirically test hypotheses that require convergence over time. As MTurk can identify distinct users, such experiments can keep an "account" or "profile" for each, giving the ability to attach information to users such as behavioral characteristics (for studying populations), model parameters (for empirical validation), or an endowment of assets (for testing economic mechanisms.) Many human computation processes also fall under the model of many users interacting over time. This type of experiment offers the ability to use longer timescales and more participants than a lab experiment but with more control and direction than a field experiment, and is uniquely suited to the online setting.

## Related Work

Several researchers have promoted the idea of using MTurk as a platform for conducting experimental behavioral research (Paolacci, Chandler, and Ipeirotis 2010; Mason and Suri 2012). They examine worker demographics and external validity, and provide guidance for common issues researchers face when using MTurk, such as building and tracking a panel of subjects, building reputation among workers, providing a "waiting room" for synchronous experiments, and checking for robustness of data. Using this methodology, Suri and Watts (2011) conducted a networked public good experiment on MTurk, and Mason and Watts (2012) tested the effects of network structure on collective learning by asking subjects to collaboratively drill for oil in a shared (artificial) territory. Both experiments were synchronous and required the interaction of multiple subjects over several distinct rounds. These unique experiments demonstrate the feasibility of MTurk as a social research platform and suggest the possibility of conducting even more powerful and expressive experiments on MTurk.

*TurKit* (Little et al. 2010b) is one of the earliest and most popular toolkits for conducting experiments on MTurk. It is specifically designed with iterative tasks in mind and makes running them on MTurk both more accessible and more powerful. However, TurKit relies on an iterative computation model and cannot handle synchronous interaction or networked communication of individuals in experiments.

In the rest of the paper, we outline the challenges of online experimentation and how we designed the architecture of TurkServer to address them and provide an infrastructure for both synchronous and longitudinal experiments. We provide a case study on using TurkServer to investigate a synchronous human computation problem, and conclude with a discussion of promising future research directions in experimental economics. We believe that the combination of these motivating problems and our experimental platform will spur new empirical research using online laboratories.

## Challenges of Online Experimentation

In the typical approach to MTurk experiments, many researchers use ad hoc methods; more time is usually spent building and debugging application code than designing experiments, and this also creates a significant roadblock for other parties to reproduce and validate existing results. Among other difficulties, one must write network code for interactive experiments, keep track of different subjects, set up appropriate experiment sessions, identify and filter noisy or bad data, and deal with many other issues in addition to designing the mechanism and interface of their experiments. We see a need for a platform that allows for efficient and practical design and data collection while reducing the obstacles to running experiments online, thus making the benefits of such experiments more readily available.

In both synchronous and longitudinal cases, the design of an experiment has two core components from a software point of view: a *user interface* defining the actions a participant can take and the information that he or she receives, and the *dynamics* or *central mechanism* that defines updates to participants and data based on actions taken or other exogenous events. From the software perspective, the distinction between the two types of experiments is only in how the users are grouped together and the length of time over which the experiment is conducted. Other capabilities of the software, such as communication, managing data, and tracking users, differ very little. Hence, the features that are helpful for conducting both types of experiments have a great deal in common:

- **Abstraction.** A primary first step for many would-be MTurk researchers is to figure out how to communicate with the MTurk SOAP or REST API, usually involving writing a nontrivial amount of code. We argue that there should be no need to interface with the Amazon Mechanical Turk API directly, or otherwise deal with excessive overhead for setting up experiments.
- **Simplified coding.** Setting up a synchronous or longitudinal experiment involves understanding the intricacies of MTurk's HIT posting and viewing mechanism, as well as writing network code for the workers to communicate with a server. Researchers are better served by focusing on implementation of the logic and interface for the experiment itself.
- **Proper Enforcement.** Spammers are a natural annoyance on MTurk, but when it comes to synchronous or longitudinal experiments, more potential issues arise: users can participate from multiple accounts and/or log into multiple HITs at once. Experimenters may also need to enforce limits on repeated session participation. These constraints should all be easy to enforce.
- **User and Data Tracking.** Experimenters need to track the participation of users, both as a recruitment tool and to evaluate performance in experiments. Results and data of experiments need to be recorded and organized. IP addresses, geolocation features, and ease of e-mailing workers allow observation and control of user demographics.
- **Noise Filtering.** In experiments with multiple participants, workers that disconnect or stop paying attention can have an especially great effect on data quality. The identity and number of these users are important in deciding if results are robust or not, and should be tracked automatically.
- **Software Support**. All of the above features must be deployable to the general MTurk population, and would be impractical otherwise. A software platform should be supported by the majority of existing workers on MTurk, yet give the experimenter the flexibility to implement necessary algorithms and features.

The TurkServer framework supports all of these commonly desired features, so that different would-be experimenters can avoid repeating the same efforts and focus on the design of experiments themselves.

## Architecture of TurkServer

TurkServer provides infrastructure for both synchronous and longitudinal experiments, resulting in a unified platform for conducting experiments using an online labor market. Researchers provide the dynamics and user interface that define the core experimental paradigm and problem, and TurkServer facilitates or even automates all other aspects. It integrates many methods that are inspired by the work of Mason and Suri (2012), and adds many features of practical significance.

TurkServer consists of server and client components, shown in Figure 1, that support bidirectional, real-time communication, and provides skeleton classes for both synchronous and longitudinal experiments. The experimenter extends the server-side classes with logic to define the dynamics of an experiment, builds a client-side user interface on top of the client component, and additionally specifies options for how users are grouped together and limits on participation. When TurkServer begins an experiment session, the server uses the MTurk API to post an appropriate number of HITs. As workers accept HITs, the user interface is loaded locally and the the client initiates a bidirectional connection to the server, using identifying metadata provided by MTurk. The TurkServer framework transparently routes messages between the experiment logic and various clients, while recording information about users and data from the experiment. In this way, both the communication between workers and the server, as well as communication with MTurk, is abstracted, and coding is specific to the experiment itself.

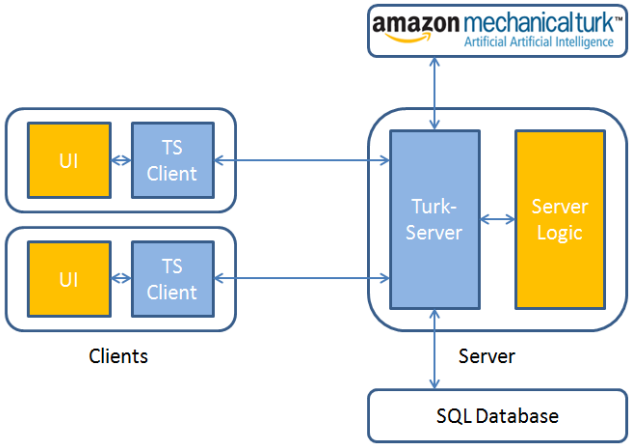


Figure 1: Basic architecture of TurkServer. The experimenter provides the components in orange by building on top of provided libraries, and the software handles communication between the various components and MTurk.

When a worker connects via the client, the server will take an action depending on the type of experiment, such as placing the worker in a virtual lobby (in synchronous experiments) or loading a worker's account information (in longitudinal experiments.) Synchronous experiments can begin rounds with a prespecified number of workers in the lobby, continuing until enough data is collected. Longitudinal experiments can do away with the lobby completely, since typically each worker interacts only with the system and not directly with other workers. Moreover, the server keeps track of the IP addresses and user IDs of workers that have accepted HITs, enforcing limits on the number of simultaneous tasks (only allowing connection from one HIT at a time, for example) and on the total number of tasks for a session. Various common (but annoying) situations, such as lost connections HITs being returned by users and picked up others, are all transparently handled.

The server uses a database to store information about

users and experiments, and supports collecting geolocation data from IP addresses, as well as customizable schemata to manage and retrieve data. The client, in addition to routing messages between the user interface and the server, monitors mouse/keyboard actions and uses a "screen saver" style mechanism to keep track of whether a user is actively participating in the experiment or has their attention elsewhere. This more "active" form of attention testing is available with real-time communication, compared to the usual test questions for non-interactive tasks such as those in Paolacci, Chandler, and Ipeirotis (2010). When a user is observed to be not paying attention, the client automatically warns them that they can be subject to prespecified penalties for not being on task, and sends the duration of inactivity to the server. TurkServer is flexible enough for other functional extensions, such as the more sophisticated "crowd instrumentation" logic as demonstrated in Rzeszotarski and Kittur (2011). The server also tracks when clients lose connection and records this information, as well as seamlessly reconnecting when clients return. These features ease the management of data and filtering of unwanted noise in experiments, and make it simpler to identify robust experimental results.

The server component, implemented in Java, allows for a great deal of implementation flexibility for the experiment dynamics, includes a self-contained webserver, and can be run on any system with a MySQL database. As for clients, two types are supported: a Java applet-based library as well as a lightweight Javascript/AJAX-based library that runs in most modern browsers. The Java-based client is more predictable to use and debug for interfaces that are complex or require more computation, but requires Java to be installed for all workers. On the other hand, the Javascript client requires some more attention to browser issues, but is very well suited for simple user interfaces and runs on just about any system.

Together, these features allow TurkServer to be used effectively not only for synchronous and longitudinal experiments, but for general experimental work where the quality monitoring, real-time communication, and database features prove useful. Indeed, the authors find it to be an effective tool for deploying many single-shot tasks on MTurk that require no user-to-user interaction. Moreover, while a great deal of the framework is devoted to abstracting away the MTurk API, the other features are useful in any experimental setting, and we plan to extend TurkServer to deploy outside of MTurk in other social computing platforms or even purpose-built websites for experimentation.

To illustrate how TurkServer can enable a complex online experiment, we describe our experience in using TurkServer to study algorithmic questions on how humans can effectively coordinate in group problem solving.

## Case Study: Algorithmic Human Computation

Human computation systems can be viewed as a new type of multiagent system, where software agents are replaced by human computers. In the future, such systems may even develop into hybrid systems that harness the respective abilities of software agents and human computers. This requires a thorough understanding of how humans interact, compete, coordinate, and collaborate with each other and with software agents. Among many questions that can be asked, an initial interesting question is the following:

> *Can successful algorithmic ideas from multiagent systems be adapted to effectively coordinate human computers to contribute jointly to a problem solving effort?*

For example, if multiple human computers were to work on a network of complex tasks, and there are constraints on the solutions between such tasks, how may we support the human computers in finding solutions that satisfy these constraints if each person is only responsible for a subset of the tasks? In one approach, we can imagine drawing inspiration from distributed coordination algorithms in multiagent systems to reach such a goal.

There are obvious difficulties and challenges. First, a human agent may not understand a complicated algorithm, and additionally needs sufficient context to effectively perform his role in the computations. In addition, the system must convey information, through good user interface design, in a way that is intuitive for humans to understand and process. Finally, humans make errors, and unless such an algorithm is somewhat robust to such errors, the performance of humans attempting to execute the algorithm can be poor. However, humans also have the ability to use instincts and make decisions in ways that a straightforward algorithm cannot, and a well-designed system should allow for this potential.

Using TurkServer, we investigate this question experimentally using a a rich set of experiments with synchronicity and real-time communication—specificially, distributed graph coloring, where agents must color the vertices of a graph such that no two adjacent vertices share the same color. Our focus on graph coloring is not because we expect human computation approaches to provide faster solutions than a computer. Rather, we view graph coloring as a paradigmatic computational problem, and one from which we hope insights will carry over to other problems. An advantage of the graph coloring problem is that non-algorithmic, human computation has been previously studied (Kearns, Suri, and Montfort 2006).

In the framework of synchronous experiments, a number of subjects are asked to color a graph with the same number of vertices. Each person can view the entire graph but controls only one vertex, and can change its color at any time. We test various adaptations of well-known *distributed constraint satisfaction* algorithms (Yokoo and Hirayama 2000). In each case, we provide normative but not fully prescriptive advice about how to participate in the task, in the spirit of the underlying algorithm. This approach centers on conveying algorithmic ideas in an intuitive way, while also allowing room for people to differ from the precise mechanistic approach.

The results of experiments like this can inform the design of human computation systems, and provide insights as to whether adaptations of user interface or instructions, using ideas from multiagent systems, can better help human solvers to complete a complex, collaborative, and interactive task.

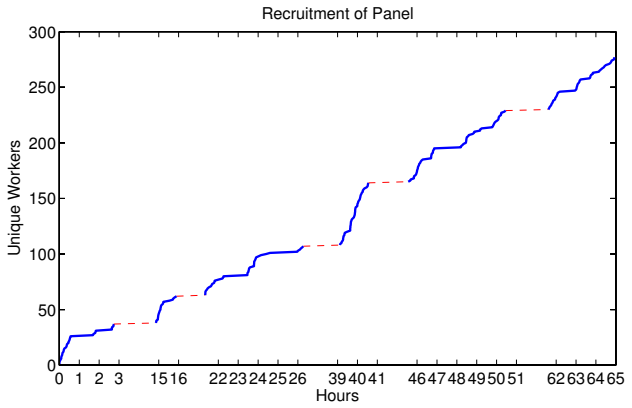


Figure 2: Recruitment of new workers over a 65-hour period, where TurkServer ran synchronous games for approximately 6 sessions spanning 24 total hours. No intervention was necessary except to start the server for each session, and almost 280 new workers were added to the database.

## Using TurkServer

To demonstrate the capabilities of TurkServer, we recruited workers and conducted experimental sessions for the graph coloring problem, and describe the process in this section. The experiments reported in this section are among the first to require multiple workers simultaneously working on the same task via MTurk, and are (to the best of our knowledge) the first utilizing real-time interaction.

Before conducting our controlled experiments (referred to as "games" for workers), we used TurkServer to coordinate numerous example games on MTurk, setting up automated recruitment sessions over several days (Figure 2). The purpose of these rounds was twofold: first, for us to test that the experiment dynamics are working as expected, and second, to build a panel of experienced workers familiar with the game. When a team of workers completed a game (either successfully or unsuccessfully), the workers were offered the choice to opt-in for messages about future sessions. TurkServer's database tracks the number of games played by each worker and other characteristics that we may want to select for a specific experiment session. Note that workers on the panel are not necessarily more successful than the average MTurk worker; they have simply demonstrated a willingness to play a full graph coloring game and have some experience playing the game. Because the recruitment sessions limit workers to few games, TurkServer recorded almost 280 unique workers over a combined period of 24 hours.

This panel of workers was used to schedule experiment sessions for collecting real data. We selected a set of workers from the panel using their opt-in or participation preferences, and TurkServer sent a message specifying the time of the experiment and providing a link to the appropriate HIT. The mechanism for these sessions is the same as the recruitment process, except that the time was announced in advance to obtain a large group of participants at once, and the limit on participation in multiple rounds is relaxed.

During each scheduled session, TurkServer continually

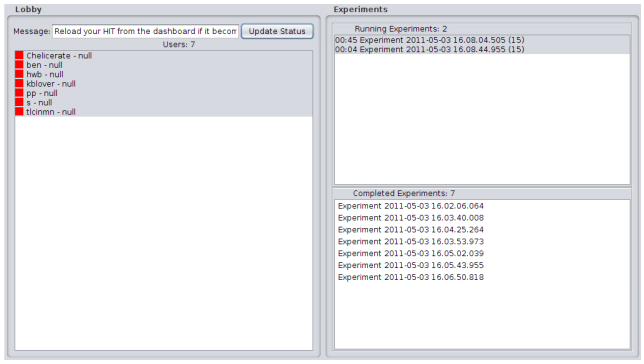


Figure 3: View of experiments from the server's perspective. The left pane shows 7 users in the lobby, and the right panes show 2 rounds (each with 15 workers) that are in progress and several others completed.

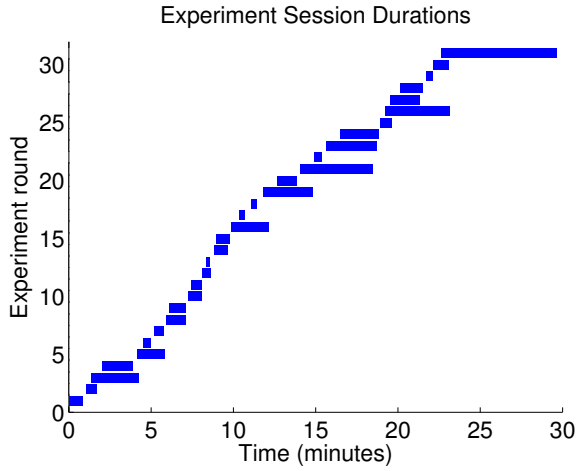


Figure 4: Start and end times for experiments in a session of 31 rounds. Each round involved 15 participants and ended when the graph was colored. Workers could only participate in one round at a time.

posts HITs to MTurk. Workers who accept a HIT join a virtual lobby and wait for a game to begin. If at any point the lobby contains enough workers, these workers are given the option to declare themselves ready for the experiment; the first fifteen workers to declare readiness then join a new experiment (and are removed from the lobby). Figure 3 shows a view of this synchronous mechanism, where users can be waiting in the lobby and multiple experiment rounds can run at once. (Additional graphical views of the experiment and users can be freely implemented by the experimenter.) Figure 4 shows the duration of 31 games from a particular session where about 60 users arrived from around a total of 200 contacted. Note that there were enough users for up to three games at once.

This particular type of experiment highlights the advantage of synchronous tasks in an online labor market. One problem that often occurs on MTurk is the prevalence of low-quality work submitted by workers. Synchronous ses-

sions have two inherent safeguards against this. First, when workers are limited to one task at a time and are paid according to the number of tasks they complete in a session, they are incentivized to achieve the goal of the task as quickly as possible. Second, when workers are aware that they are collaborating or competing with other humans, we notice that the social aspect greatly increases interest compared to normal single-interaction tasks.

Nevertheless, TurkServer has features to monitor workers' activity during an experiment and ensure that they were actually participating. In this particular case, when a worker does nothing (including mouse movements) for 45 seconds, an inactivity warning replaces the game on their screen using the screen saver mechanism. Workers were warned that being inactive for an unspecified amount of time would result in not being paid for the HIT. In all of our scheduled sessions, we did not find that worker inactivity was a significant issue. The majority of workers never saw the warning, and for those that did, they immediately moved their mouse.

We find that the ideal rate to pay for these experiments is somewhat higher than the average going rate on MTurk. Primarily, we want motivated users to earn more in an experiment session than they could doing any other task, so that they are not only willing to participate in our sessions, but will also exert greater effort as their pay is commensurate with the number of tasks completed. In these particular experiments discussed, workers were paid a constant rate of $0.20 per game, with no bonus. We were able to carry out 30 games of 15 participants each in a session lasting around half an hour (Figure 4), at a cost of less than $100 per session.

## Discussion and Future Work

TurkServer is certainly not limited to human computation experiments, but opens many potential new and exciting directions for empirical work. We conclude with a discussion potential experiments on asset and prediction markets that also fall within the synchronous and longitudinal setting, and can be very informative for both experimental economics and collective intelligence.

Markets play a central and irreplaceable role in regional and global economies. The ubiquity of modern financial markets can be partially attributed to the theoretical efficiency of market mechanisms, that is, the ability of markets to quickly incorporate all information relevant to traded assets into prices (Fama 1970). The importance of markets has led to many empirical studies in the field of experimental economics, including the seminal experimental work on dissemination and aggregation of private information of traders in a market (Plott and Sunder 1982; Sunder 1995). However, as far as we are aware, no such real-time trading experiments have been conducted on platforms like MTurk.

In the last decade, the idea of informational efficiency has also motivated the development of prediction markets, specifically designed for eliciting and aggregating information relevant to some event of interest and generating an accurate forecast. New market mechanisms, such as logarithmic market scoring rules (Hanson 2003; 2007) and combinatorial market mechanisms (Abernethy, Chen, and Vaughan 2012), have been designed for prediction markets to better achieve the goal of information aggregation. New theories have been developed to understand conditions for information aggregation and impacts of trader manipulation in markets (Ostrovsky 2009; Chen et al. 2010; Iyer, Johari, and Moallemi 2010), with very little empirical verification, except (Hanson, Oprea, and Porter 2006; Jain and Sami 2010). The market framework has also been generalized to eliciting answers and other contributions in crowdsourcing (Abernethy and Frongillo 2011).

We see promise in conducting market experiments in an online setting, in exactly the setting that they will be used—with human traders participating in a noisy fashion connected by the Internet—yet in a controlled manner. In particular, we envision three interesting types of experiments.

First, we can imagine extending classical laboratory experiments to allow for many more possibilities. Such experiments typically have 6 to 12 participants with very specific information structures (e.g. half of the participants know the value of an asset with certainty and the other half have no information), offer artificial assets, and have "markets" that often last for 3 minutes. We can imagine a variety of settings with more participants and more complex signal structures to examine the robustness of classical results.

In addition, we suggest investigating recently proposed mechanisms to compare their performance on disseminating and aggregating information with that of the well-studied continuous double auction. Moreover, this allows for experimentally testing and validating recent theories on information aggregation and manipulation in markets.

Finally and perhaps most interesting, we propose longitudinal prediction market experiments for real world events (e.g. the U.S. general presidential election). There are several real prediction markets for political, economic, and sporting events, such as Intrade and the Iowa Electronic Markets, predicting events that will happen in the real world with open trading for months. Using data from these markets, researchers have empirically studied prediction market accuracy. However, it is generally impossible to view the transactions of individual traders and study their strategic behavior. With a longitudinal experimental market, we can control the number of participants and track the actions of each individual trader, allowing for a closer examination of market dynamics and strategies of traders.

Using the infrastructure and features provided by TurkServer, we provide the community with a powerful toolbox that enables more expressive and interactive experiments online, with features that streamline participant recruitment, experiment implementation, and data collection. As the vast majority of social experiments fit into the synchronous or longitudinal paradigm that we discussed, we imagine that TurkServer will be of great use to both the human computation and social science community. With extensive online experimental research, empirical work can more quickly catch up with theory, and inform the state of the art in experimental social science, collective intelligence, social computing, and many other fields.

**Acknowledgements** We are grateful to Sid Suri for invaluable discussions on online behavioral experiments. This material is based upon work supported by NSF Grant No. CCF-0953516 and Xerox Foundation. Haoqi Zhang is generously funded by a NSF Graduate Research Fellowship. In addition, research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

**Online Resources** TurkServer is currently open source[1] and in active development. We encourage readers who are interested in the software to contribute ideas or code that can make it more useful to the community. As the API for TurkServer is subject to change, we have not included technical details in this paper; please visit the link below for the most current information.

## References

Abernethy, J., and Frongillo, R. 2011. A collaborative mechanism for crowdsourcing prediction problems. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS)*.

Abernethy, J.; Chen, Y.; and Vaughan, J. W. 2012. Efficient market making via convex optimization, and a connection to online learning. *ACM Transactions on Economics and Computation*. To Appear.

Amir, O.; Rand, D. G.; and Gal, Y. K. 2012. Economic games on the internet: The effect of $1 stakes. *PLoS ONE* 7(2):e31461.

Bernstein, M. S.; Brandt, J.; Miller, R. C.; and Karger, D. R. 2011. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th Symposium on User Interface Software and Technology (UIST)*.

Chen, Y.; Dimitrov, S.; Sami, R.; Reeves, D. M.; Pennock, D. M.; Hanson, R. D.; Fortnow, L.; and Gonen, R. 2010. Gaming prediction markets: Equilibrium strategies with a market maker. *Algorithmica* 58(4):930–969.

Fama, E. 1970. Efficient capital markets: A review of theory and empirical work. *Journal of Finance* 25(2):383–417.

Fishbacher, U. 2006. z-tree reference manual.

Hanson, R.; Oprea, R.; and Porter, D. 2006. Information aggregation and manipulation in an experimental market. *Journal of Economic Behavior & Organization* 60(4):449–459.

Hanson, R. 2003. Combinatorial information market design. *Information Systems Frontiers* 5(1):105–119.

Hanson, R. 2007. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets* 1(1):3–15.

Horton, J. J.; Rand, D. G.; and Zeckhauser, R. J. 2011. The online laboratory: Conducting experiments in a real labor market. *Experimental Economics* (14):399–425.

Iyer, K.; Johari, R.; and Moallemi, C. C. 2010. Information aggregation in smooth markets. In *Proceedings of the 11th ACM Conference on Electronic Commerce (EC)*, 199–205.

Jain, L., and Sami, R. 2010. Aggregation and manipulation in prediction markets: Effects of trading mechanism and information distribution. In *Proceedings of the 10th ACM Conference on Electronic Commerce (EC)*, 207.

Kearns, M.; Suri, S.; and Montfort, M. 2006. An experimental study of the coloring problem on human subject networks. *Science* 313:824–827.

Little, G.; Chilton, L. B.; Goldman, M.; ; and Miller, R. C. 2010a. Exploring iterative and parallel human computation processes. In *Proceedings of the 2nd Human Computation Workshop (HCOMP)*.

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C. 2010b. Turkit: Human computation algorithms on mechanical turk. In *Proceedings of the 23rd Symposium on User Interface Software and Technology (UIST)*.

Mason, W., and Suri, S. 2012. Conducting behavioral research on Amazon's Mechanical Turk. *Behavior Research Methods* 44(1):1–23.

Mason, W., and Watts, D. J. 2012. Collective problem solving in networks. *Proceedings of the National Academy of Sciences*. in press.

Ostrovsky, M. 2009. Information aggregation in dynamic markets with strategic traders. In *Proceedings of the 10th ACM Conference on Electronic Commerce (EC)*, 253.

Paolacci, G.; Chandler, J.; and Ipeirotis, P. 2010. Running experiments on amazon mechanical turk. *Judgment and Decision Making* 5(5):411–419.

Plott, C. R., and Sunder, S. 1982. Efficiency of experimental security markets with insider information: An application of rational-expectations models. *Journal of Political Economy* 90(4):663–98.

Rand, D. G. 2012. The promise of mechanical turk: How online labor markets can help theorists run behavioral experiments. *Journal of Theoretical Biology* (299):172–179.

Rzeszotarski, J. M., and Kittur, A. 2011. Instrumenting the crowd: using implicit behavioral measures to predict task performance. In *Proceedings of the 24th Symposium on User Interface Software and Technology (UIST)*.

Shahaf, D., and Horvitz, E. 2010. Generalized task markets for human and machine computation. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 986–993.

Sunder, S. 1995. *Experimental asset markets: A survey*. Princeton NJ: Princeton University Press. chapter 6, 445–500.

Suri, S., and Watts, D. J. 2011. Cooperation and contagion in web-based, networked public goods experiments. *PLoS ONE* 6(3):e16836.

von Ahn, L., and Dabbish, L. 2008. Designing games with a purpose. *Communications of the ACM* 51(8):58–67.

Yokoo, M., and Hirayama, K. 2000. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems* 3:185–207.

Zhang, H.; Law, E.; Miller, R. C.; Gajos, K.; Parkes, D.; and Horvitz, E. 2012. Human computation tasks with global constraints. In *Proceedings of the 30th ACM Conference on Human Factors in Computing Systems (CHI)*.

[1] http://turkserver.googlecode.com