The Complexity of Crowdsourcing: Theoretical Problems in Human Computation

Anand Kulkarni
Department of Industrial Engineering and
Operations Research
4141 Etcheverry Hall, University of California,
Berkeley, 94720-1777
anandk@berkeley.edu

ABSTRACT
What does theoretical computer science have to say about human computation? We identify three problems at the intersection of crowdsourcing, operations research, and theoretical computer science whose solution would have a major impact on the design, evaluation, and construction of real crowdsourcing systems. In some cases, these problems can let us sidestep apparently difficult HCI challenges by making appropriate choices at the algorithmic level. In other contexts, theoretical tools provide a formal basis for evaluating the performance of algorithms and classifying the difficulty of tasks in crowdsourcing. Our problems are illustrated through two recent projects. The first, *Turkomatic*, is an attempt to construct a "universal" algorithm for generating workflows on microtask crowdsourcing platforms. The second, *MobileWorks*, is a new crowdsourcing engine designed from the bottom up to provide appropriate abstractions between the theoretical elements of human computation systems and interface/design questions. It is hoped that this analysis can spur the development of theoretical frameworks for understanding algorithms involving human computation.

Author Keywords
Crowdsourcing, foundations, algorithms, complexity

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Theory, Algorithms, Human Factors

INTRODUCTION

Jeanette Wing, director of the NSF computing division, has recently suggested [1] that the development of theoretical computer science tools for systems involving human computation is one of the five most important questions facing computing today. These tools have had a profound impact in the development of computer science, but their development is overdue in crowdsourcing: in both conventional and quantum computing, formal models of computation and algorithms preceded the development of reliably working systems by decades. In human computing, experimental progress in building crowdsourcing systems has dramatically outpaced the development of theoretical models. How can human computation benefit from the set of tools theory offers?

As an example of a benefit theoretical frameworks can bring to crowdsourcing, consider that computer science has long enjoyed established models of computation for evaluating and comparing the performance of algorithms independent of the specifics of their implementation. However, no such abstract models for comparison exist in human computation. For example, how should we compare techniques for solving a problem using Games with a Purpose [2] against ones powered by Amazon's Mechanical Turk [3]? Current analyses typically report on measures that are heavily influenced by the underlying microtask platform, implementation and user interface (monetary cost, user engagement, conflating platform-specific features of success with independent, generalizable advantages of the algorithm or approach used. As the number of platforms available for crowdsourcing increases, this problem will only get worse.

One solution is to build a model of computation involving humans. A recent model proposed by Shahaf and Amir suggests extension of the Turing Machine model to include calls to a human computation engine (ie, an oracle) [4]. Under this model, we can compare the cost of two human computation algorithms as a weighted sum of the number of operations and the number of yes-or-no oracle queries required. We argue that this is the appropriate level of abstraction in considering algorithms that use human computation. First, it completely separates error control, user rewards, and user interface questions in crowdsourcing evaluations from algorithm design questions. Second, it allows us to directly quantify the cost of systems using human computation as the number of queries made to an oracle, irrespective of the specific underlying ways that humans are being rewarded or recruited into a system. Third, it lets us give an objective theoretical comparison between algorithms using humans and the best known algorithms that do not use crowds. This last advantage may even mean that problems solved using human computation may have a role in the existing hierarchies of computational complexity, much as randomized and quantum complexity classes have been added to these hierarchies.

We mention this extension of the Turing Machine model as a first example of a potentially rich intersection between CS theorists and scientists in HCI in developing a formal theory of crowdsourcing. In the remainder of the paper, we discuss three problems in human computation that the author has pursued that emerge from theoretical computer science and operations research: automatic workflow design, hierarchical system design, and models for real-time computation.

TURKOMATIC: ALGORITHMS FOR AUTOMATIC TASK DECOMPOSITION AND WORKFLOW DESIGN

A central question of interest in human computation is the *workflow design* or *task decomposition* problem.

Workflow Design: Given an arbitrary high-level task, how can we break it down into a workflow of tasks that can be fed into a human computation platform?

Processing complex tasks on crowdsourcing platforms like Mechanical Turk currently requires substantial up-front investment by designers into task decomposition and workflow design. Historically, the problem of workflow design has been approached in an ad hoc manner, with individual workflows designed by system creators at substantial up-front cost. Treating this problem as a computational one, it is reasonable to ask whether we can design algorithms to automatically construct workflows for tasks given as input.

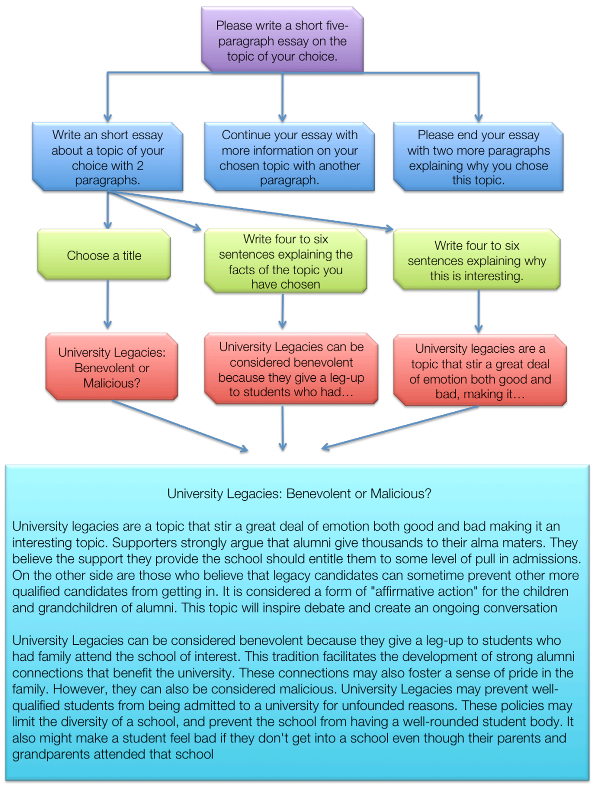
In joint work with Matthew Can and Bjoern Hartmann, the author has proposed a new method for automating task and workflow design for high-level, complex tasks [5]. We suggested problem of high-level task design could be partly automated by assigning the responsibility of designing workflows to workers themselves – a *recursive* algorithm for workflow design.

This algorithm was implemented on Mechanical Turk. This system, Turkomatic, generates Human Intelligence Tasks (HITs) asking Mechanical Turk workers (Turkers) to decompose complex tasks into simpler ones, solve these tasks in parallel, and combine the results into a coherent solution.

Turkomatic's interface accepts a description of a general task from the end user posed in natural language and posts a HIT asking workers to break the task down into a set of logical subtasks. These subtasks are automatically reposted to Mechanical Turk, where workers can choose to simplify them further — a recursive subdivision — or to solve them directly. Once all subtasks are completed, HITs are posted asking workers to combine subtasks' solutions into a coherent whole, which is returned to the requester.

Our view is that this approach can be tremendously powerful in extending the kinds of problems solvable via human computation. Because designing effective tasks on crowdsourcing systems like Mechanical Turk is currently a black art, it is unknown how to convert the kind of open-ended high-level tasks people and organizations do every day ("write a paper about ____; build a webpage containing ____; build software that does ___") into tasks that can be solved effectively on crowdsourcing systems.

However, in recent experiments, we have used Turkomatic algorithm to automatically decompose complex, multistage tasks like essay-writing into small pieces (Fig 1). Implemented effectively, Turkomatic can serve as a kind of universal solver for tasks on Mechanical Turk, although it is


Please write a short five-paragraph essay on the topic of your choice.
Write an short essay about a topic of your choice with 2 paragraphs.
Continue your essay with more information on your chosen topic with another paragraph.
Please end your essay with two more paragraphs explaining why you chose this topic.
Choose a title
Write four to six sentences explaining the facts of the topic you have chosen
Write four to six sentences explaining why this is interesting.
University Legacies: Benevolent or Malicious?
University Legacies can be considered benevolent because they give a leg-up to students who had…
University legacies are a topic that stir a great deal of emotion both good and bad, making it…

University Legacies: Benevolent or Malicious?

University legacies are a topic that stir a great deal of emotion both good and bad making it an interesting topic. Supporters strongly argue that alumni give thousands to their alma maters. They believe the support they provide the school should entitle them to some level of pull in admissions. On the other side are those who believe that legacy candidates can sometime prevent other more qualified candidates from getting in. It is considered a form of "affirmative action" for the children and grandchildren of alumni. This topic will inspire debate and create an ongoing conversation

University Legacies can be considered benevolent because they give a leg-up to students who had family attend the school of interest. This tradition facilitates the development of strong alumni connections that benefit the university. These connections may also foster a sense of pride in the family. However, they can also be considered malicious. University Legacies may prevent well-qualified students from being admitted to a university for unfounded reasons. These policies may limit the diversity of a school, and prevent the school from having a well-rounded student body. It also might make a student feel bad if they don't get into a school even though their parents and grandparents attended that school

Figure 1: Turkomatic's recursive algorithm automatically generates crowdsourcing workflows for complex, high-level tasks. Here, the result of giving Turkomatic the task, "Write a five-paragraph essay."

plainly less efficient in the number of humans used than workflows tailored by an expert. Our continued experiments with Turkomatic and understanding how it fails will shed light on the formal complexity of the workflow design problem.

MOBILEWORKS: MAKING HUMAN COMPUTATION CONSISTENT WITH INCONSISTENT WORKERS

The model of humans as infallible oracles suggested by the human-assisted Turing Machine model is in dramatic contrast to the way humans actually behave on crowdsourcing platforms – they are unreliable, inconsistent, and often wrong. This is not as problematic as it first appears; after all, in conventional computing, hardware errors have not been eliminated but tools for designing software no longer need to concern themselves with these issues. In the author's view, human computation must move in a similar direction to enable more efficient programming. In particular, this involves resolving three problems:

Consistency: Given the same inputs, how can we get the same output from a microtask platform, independently of the time of day?
Speed: How can we get results from a microtask platform with a predictable speed?
Accuracy: How can we predict the accuracy of a query given to a microtask platform in advance?

These three questions are equally amenable to solution through improved interface design or alternative algorithmic choices. Consider, for instance, the accuracy problem. Redundancy and majority voting, by far the most common methods in use today, are only the simplest techniques that have been developed in literature on social choice and voting algorithms. To the extent that more sophisticated tools are unknown in the crowdsourcing community, it will be important to survey these techniques and identify which are best suited for particular classes of problems.

The difficulty of the accuracy question is illustrated in MobileWorks, an ongoing project by the author, Philip Gutheim, Prayag Narula, and Dave Rolnitzsky. MobileWorks is a social enterprise and crowdsourcing platform letting low-income workers in the developing world participate in the crowdsourcing economy through their mobile phones. Our worker pool consists of mobile-phone owners in developing nations living on less than $2 a day. The MobileWorks interface accepts handwritten documents, divides them into components, and sends each piece to workers to be solved as OCR; they are paid higher-than-typical wages for the task in an effort to lift them out of poverty. Because the project's objectives are social, the worker pool is unusually unreliable – many come from low-education backgrounds and historically marginalized groups.

As an effort to provide employment to a semiliterate population, MobileWorks needs to reconcile the inherent unreliability of its worker pool with the need to provide efficient solutions. Because OCR work operates on extremely slim price margins, the problem of eliminating worker error cannot be addressed through redundancy in the worker pool alone. We chose to balance a small number of expensive, redundant checks carried out within the worker pool with a combination of periodic qualifications and reputation tracking, as well as the ability to occasionally send work to a cheaper alternative platform entirely for error control (in this case, Mechanical Turk). This comprehensive all-of-the-above approaches works practically for getting crowdsourcing systems to function reliably. However, as is common for these strategies, it is not obvious how to most efficiently combine these techniques to minimize costs.

As a result, MobileWorks represents an instance of a project where additional theoretical modeling and cost optimization can make a substantial difference in achieving the platform's objectives.

MODELS FOR REAL-TIME HUMAN COMPUTATION

There are presently few practical systems that make use of real-time human computation. Part of this reason is historical: because they were intended for large-scale data processing tasks, microtask marketplaces like Mechanical Turk are largely designed for asynchronous participation.

In the past six months we have seen the emergence of applications that make use of online or nearly-real-time responses, such as VizWiz [6] and Soylent [7]. These are a compelling class of applications with strong potential for bringing the benefits of crowdsourcing out of cloud data-processing and into end-user applications. Of particular interest to the author is the potential for improving robotic decision-making and human-robot communication, and look forward to building robots whose intelligence can be extended in real time by contributions from the crowd.

However, before these systems can be built, we need to determine ways to make systems using human computation behave quickly. In joint work with Siamak Faridani and Kuang Chen, the author is designing models to represent the time required to produce solutions to tasks posted on microtask markets. Our approach is based on infinite-server queuing models in queuing theory, which represent the relationship between arrival of users (servers) at crowdsourcing websites and the arrival and processing of jobs.

As part of this work, we are attempting to build a predictive system that can accept parameters describing any crowdsourcing platform and particular set of problem requirements, and determine automatically when and how task should be posted, including pricing and task size, to make sure that the task is completed by a certain time with a theoretically derived probability. This presents an alternative approach to solutions such as QuikTurKit [6], which use novel worker interaction techniques to obtain faster response times. Our earliest predictions from a purely theoretical analysis of our model are summarized below.

There exist critical lower- and upper- density for real-time responses. Adding additional workers to a microtask platform decreases response time for a task, but only until a critical threshold of workers is reached; beyond this number, no additional workers joining the system increase its ability to process a task faster.

There exists a critical reward threshold beyond which additional payments do not increase completion speed in a microtask market. Increasing rewards (payments on Mechanical Turk, or points in a game) improves completion speed, but only until particular thresholds are reached – our model shows that additional rewards beyond a certain point will not decrease the time required for a task to be completed.

CONCLUSION

Our vision for human computation is to see the field develop the kind of rigorous foundation and abstractions that exist in classical autonomous computing, especially models and techniques will enable researchers to better build and reason about these systems.

Human computation offers a rich vein for CS theorists to answer new, practical questions and come up with alternative, algorithmic-driven solutions to problems that have so far been addressed as interface challenges. We do not mean to suggest that the list of theoretical questions we present is exhaustive – it is simply the beginning of a potentially rich examination of problems at the intersection of theory and human computation. The resolution of each of these issues can begin to form the basis of a theory of human computation.

ACKNOWLEDGMENTS

Thanks to the collaborators on the projects discussed for many useful discussions: Bjoern Hartmann and Matthew Can on Turkomatic, Siamak Faridani and Kuang Chen on real-time modeling, Prayag Naurla, Philipp Gutheim, and Dave Rolnitzky on MobileWorks, and Steven Dow and Bjoern Hartmann in general.

Author Biography

Anand Kulkarni is a 5th-year PhD student in Industrial Engineering and Operations Research at the University of California, Berkeley and an NSF Graduate Research Fellow. Although his training and background are in theoretical computer science, mathematics, and algorithms, he is interested in how ideas from these fields can inform and extend practices in crowdsourcing.

Anand co-organizes Berkeley's graduate research seminar on crowdsourcing, and his research concerns applications at the intersection of crowdsourcing and theory, crowdsourcing and robotics, and crowdsourcing and international development. He holds Bachelor's degrees in mathematics and physics and a Master's degree in operations research at UC Berkeley.

REFERENCES

1. Wing, J. Five deep questions in computing. Comm. ACM, Vol. 51, No. 1. (2008), pp. 58-60
2. von Anh, L, and Dabbish, L. Designing Games with a Purpose. Communications of the ACM 51, 8 (Aug. 2008), p. 58-67.
3. Amazon's Mechanical Turk. http://www.mturk.com
4. Shahaf, D., and Amir, E.. Towards a theory of AI-completeness. Proceedings of Commonsense 2007, 8th International Symposium on Logical Formalizations of Commonsense Reasoning.
5. Kulkarni, A., Can, M., Hartmann, B. Turkomatic: Automatic Recursive Task and Workflow Design for Mechanical Turk. Submitted, CHI2011 WIP.
6. Bigham, J., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R.C., Miller, R., Tatrowicz, A., White, B., White, S., and Yeh, T.. VizWiz: Nearly Real-time Answers to Visual Questions (2010). In Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2010).
7. M. Bernstein, G. Little, R.C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, K. Panovich, Soylent: A Word Processor with a Crowd Inside, Proceedings of UIST 2010.