Human Computation 1 (2014) 569–592
<Copyright and creative commons notice>
ISSN: 2330-8001, DOI: <Insert DOI here>

Toward Complexity Measures for Systems Involving Human Computation

R. JORDAN CROUSER, TUFTS UNIVERSITY

BENJAMIN HESCOTT, TUFTS UNIVERSITY

REMCO CHANG, TUFTS UNIVERSITY

ABSTRACT

This paper introduces the Human Oracle Model as a method for characterizing and quantifying the use of human processing power as part of an algorithmic process. The utility of this model is demonstrated through a comparative algorithmic analysis of several well-known human computation systems, as well as the definition of a preliminary characterization of the space of human computation under this model. Through this research, we hope to gain insight about the challenges unique to human computation and direct the search for efficient human computation algorithms.

1. INTRODUCTION

Computational complexity theory is a branch of theoretical computer science dedicated to describing and classifying computational problems according to their fundamental difficulty, which we measure in terms of the resources required to solve them. One way to measure a problem's difficulty is with respect to time; we may ask **how many operations do I need to perform to find an answer**? Alternatively, one might want to measure difficulty in terms of space; here we could ask **how much memory will I need to execute this process**? These questions, which do not rely on the specific implementation details, are at the heart of computer science. Theoretical arguments ground our intuitions about the problem space, and pave the way for us to design future systems that make these provably correct solutions tractable.

To date, the field of human computation has concerned itself almost entirely with questions of *tractability*. That is, can using human computation make it possible to efficiently solve problems whose solutions are otherwise thought to be too expensive to compute? Using experiential knowledge regarding the kinds of processing that humans are "better" at, such as recognizing objects and speaking naturally, we build systems that capitalize on these skills and offer them as constructive proof: tangible evidence that the problems are in fact tractable using human computation, even when other methods have failed. As such, the development of real-world implementations has far outpaced the development of theoretical measures. Many of these implementations have demonstrated unparalleled success at problems previously thought to be intractable, such as protein folding (Cooper et al., 2010). However, in the absence of a rigorous theory in which to ground new algorithms, researchers must rely on intuition and some deeply-rooted assumptions about the differences between human and machine computation. Extending complexity models to encompass both human and machine computation will help us understand our successes and failures on existing problems, and more fully illuminate the computational space they inhabit.

Computational complexity theory takes the study of solvable problems to a deeper level by asking about the **resources** needed to solve them in terms of time and memory. It enables us to ask questions that get at the fundamental nature of the problem and how we might go about solving it more effectively. Does randomization help? Can the process be sped up using parallelism? Are approximations easier? By understanding the resources required, we can begin to group problems into *complexity classes*, with members of the same class requiring similar kinds or quantities of resources. It also enables us to investigate the effect of limiting these resources on the classes of tasks that can still be solved.

Until now, no such theoretical apparatus has existed for human computation. This is due in large part to the fact that our ability to model **how** the human brain computes is hindered by a limited understanding of the biological mechanisms that enable that computation. While cognitive modeling techniques can help us to understand the interplay between stimulus and response, existing modeling tools are not designed to emulate the *complexity* of the model itself. Until our understanding of the cognitive processes involved in computation is more fully developed, it seems likely that human computation will generally remain a (somewhat finicky) black box in the larger system diagram. In the interim, we can begin to to characterize and quantify the use of human processing power as part of an algorithmic process, rather than waiting to model and measure the cost of the humans' computational processes themselves. By separating questions of per-operation cost from questions of resource utilization, we posit that such models will be robust even as more nuanced and complete models of the human brain come to light.

In this work, we introduce the notion of Human computational complexity theory - an extension of traditional computational complexity theory that accounts for (some of) the vagaries human participatory behavior. We present the Human Oracle Model as a novel method for characterizing and quantifying the use of human processing power as part of an algorithmic process. We then use this model to compare and analyze several well-known human computation systems in the area of Image Labeling, which serves as a vignette to demonstrate the kinds of low-level algorithmic comparisons which might be possible under an abstract model. Finally, we describe how this model can be used to characterize the space of human computation, as well as discuss the model's limitations and its potential for broader impact.

We argue that in order for human computation to achieve legitimacy as a breakthrough computational paradigm, it is imperative that we as a community establish mechanisms by which we abstract the underlying processes away from the details of the implementation and *reason* about the computation being done in human computation systems. The examples presented herein are just a small scratch on the surface of that much larger problem. Their purpose is not to exhaustively illuminate the landscape of human computation problems, but to illustrate that such reasoning is possible using only minor modifications to tools with which computational scientists are already familiar. Through this research, we hope to gain insight about issues unique to this problem space and direct the search for more efficient human computation solutions. In addition, we hope to set the stage for continued dialogue between the human and traditional computation communities, that together we may move forward in pushing the bounds of what is possible and challenge our deep-seated assumptions about what it means to compute.

2. FOUNDATIONS AND RELATED WORK

In traditional computer science, the field of **computational complexity** compares computational processes by evaluating the time and space required to solve a well-specified problem using a computer (for a detailed introduction to computational complexity, please see (Arora and Barak, 2009)). In many cases, it is particularly useful to consider the boundary cases of such problems – that is, what is the best we can hope to do under ideal conditions, and what kind of performance can be guaranteed in the worst case? By better understanding these boundary cases, we become able to characterize the practical limits of what computational processes can do, as well as better isolate the underlying subtasks that make some computational problems truly difficult to solve. When theoretical modeling is applied to systems that involve humans, we posit that it is similarly useful to begin by establishing bounds on clearly-defined edge cases, and we can remain agnostic to behaviors for which boundaries are currently ill-defined so long as we understand how such behavioral processes affect overall system behavior in the limit.

Under existing computational models, many interesting real-world problems are known to be computationally infeasible, even if the path to finding the solution is clear. For example, we know how to solve the Traveling Salesman problem, but computing the solution is intractable for all but a handful of special cases (Lawler et al., 1985). Other problems, such as general image recognition, have no known computational solution at all. In contrast, humans appear to perform remarkably well on many such problems with relative ease (MacGregor and Ormerod, 1996). Robust perceptual systems and learned experiences offer humans a distinct advantage in areas such as visual classification, a fact which lies at the heart of the emerging field of human computation. While our understanding of the biological mechanisms that enable computation in the human brain is still limited, there is an intuition that human computational processes are different from, and in may cases complementary to, mechanical computation.

One recent framework (Crouser and Chang, 2012) illustrates this complementarity, and attempts to organize existing literature on human-machine collaborative systems according to which skills, or affordances, the system leverages. Other taxonomies (Bertini and Lalanne, 2010; Quinn and Bederson, 2011) propose additional classification dimensions such as *human-machine balance*, *motivation*, *aggregation*, *quality control*, *process order*, and *task-request cardinality*. While these frameworks provide a vocabulary for describing human-computer collaborative systems, they fall short of enabling us to quantify the computational work being done in the underlying human computation algorithms. Consider a sample of the numerous published human computation systems in the area of Image Labeling: the ESP Game (von Ahn and Dabbish, 2004), KissKissBan (Ho et al., 2009) and Peekaboom (von Ahn et al., 2006). Each employs human visual perception and linguistic ability to process and describe images, and uses the machine to distribute tasks and aggregate the results. Each uses entertainment as a primary motivator, and redundancy to ensure validity of the resulting data. Given the similarity of the problem as well as the approach to solving it, how do the underlying algorithms compare? To date, there exists no mechanism for performing such a comparison.

Establishing bounds on algorithmic processes and deepening our understanding of the relationships among the problems they solve are of critical importance to the study and design of systems involving human computation. Drawing parallels at the algorithmic level rather than at the implementation level enables us to compare solutions more rigorously than using simple A-B testing. As with other branches of computational science, identifying areas where existing algorithms are redundant or inefficient will enable the design of more efficient algorithms in the future. In addition, reporting bounds on the complexity of human computation algorithms along with the observed performance of the system would improve study reproducibility, as well as help isolate the effects of interface design and other implementation details.

In order to facilitate these kinds of comparisons, it is important to extend traditional theoretical abstractions of computation to capture some of the characteristic differences between computing on silicon and computing using human brainpower. This is a daunting challenge: precisely which facets of "humanness" are the critical components which must be captured in order for a computational model to be both robust and useful? We intentionally err on the side of simplicity in this early attempt, looking to existing work in measuring intelligent contribution to a computational process from the field of Artificial Intelligence as a foundation and asking what we might learn about human computation from even the simplest of theoretical abstractions. While we most certainly fall short of solving the problem of measuring the complexity in human computational systems, it is our hope that this preliminary work will provoke both thought and debate about the role of computational complexity in human computation, and vice versa.

2.1. Computation with Human Oracles

Research in the field of Artificial Intelligence seeks to model and emulate human intelligence using a machine. Research in human computation leverages *actual* human intelligence to perform computationally-difficult tasks. Both fields hinge on the long-held belief that there exist problems that require human-level intelligence and reasoning to solve. Because of this relationship, we believe that theoretical models from the Artificial Intelligence community may be a useful starting point for understanding and comparing human computation problems and their solutions.

Theoretical computer science uses abstract models of computational systems, such as Turing Machines (Turing, 1938), to simulate computational processes[1] and explore the limits of what can be

[1]We have elected to begin by expanding on the standard notion of the Turing Machine in this early work due to its elegance, completeness, and relative ubiquity in the discussion of classical computational complexity. It has been suggested that other models, such as non-deterministic Turing Machines, Markov Chain models or Abstract State Machines more generally, might also be useful in exploring the edges of human computation. We hope that the utility of these and other models will be explored in future work.

computed. In some cases, it is useful to allow the Turing Machine to query an Oracle – a super-powerful subroutine which is able to decide specific problems with perfect accuracy in constant time. Under this model, the overall cost of running the algorithm is the machine's computation plus the time required to generate queries to the Oracle; the Oracle's computation is performed for free. While the computational machinery used by the Oracle to perform its computation is a black box, we can characterize the amount of work the Oracle is being asked to perform by counting the number of queries the Oracle must answer in order for the computation to terminate. This is known as *query complexity*.

Shahaf and Amir proposed an extension to the standard computational model in which questions may be asked of a Human Oracle – an Oracle with human-level intelligence (Shahaf and Amir, 2007). In this model, the Human Oracle is able to answer questions to which a human would be able to respond, even if a machine could not. They suggest that the complexity of an algorithm executed on such a machine can be represented as a pair $\langle \Phi_H, \Phi_M \rangle$, where Φ_H indicates the query complexity of the algorithm (number of queries to the Human Oracle) as a function of the input size, and Φ_M is the the complexity of the computation performed by the machine. Whenever the complexity of the machine's computation is the same, the complexity of two algorithms can be compared by considering which has a higher query complexity. The minimal complexity of a problem can then be thought of as the minimization of both human and machine cost over all algorithms that correctly solve the problem.

2.2. Value of an Oracle Model for Human Computation

Modeling the human contributions to an algorithm as queries to an Oracle captures the underlying behavior of many existing human computation algorithms. For example, in the well-studied ESP Game (von Ahn and Dabbish, 2004) a human is given some input (an image) and, like an Oracle, is expected to provide a (relatively) correct response to exactly one question: *What do you see?* This interchange, where an external entity is used to inexpensively perform some challenging subroutine, is exactly the kind of system that Oracle machines were designed to describe. Because of this, we adopt the Human Oracle Model as a preliminary mechanism to make quantitative comparisons among human computation algorithms.

Despite the simplicity of the Human Oracle Model, this level of abstraction has several benefits. First, it enables a direct quantification of the cost of an algorithm leveraging human-level intelligence, or human computation, in terms of the number of queries made to the human. This enables a straightforward comparison between two human computation solutions to a given problem on a given input. Second, it enables an objective theoretical comparison between algorithms using humans and the best known purely mechanical algorithms, if they exist. Finally, it separates implementation-specific details such as error control, motivation, and interface design from the algorithm itself. This is an important differentiation, and much in keeping with the spirit of traditional complexity models wherein the performance of an algorithm is assessed independent of the languages in which it may later be implemented or the hardware on which it may be run. While questions of recruiting and incentivizing human contributors is by no means unimportant, we specifically investigate the complexity of the underlying algorithms independently.

Technically speaking, a human can simulate any process the machine can execute. Given an understanding of the process, enough paper and a sufficient supply of pencils, a human operator could write out the contents of each register, perform each bitwise operation, and record each result by hand. However, the time and resources required by the human to compute exactly the same result would be exorbitant. In addition, humans are susceptible to fatigue, and are arguably limited by unreliable recall and the capacity of working memory. In this sense, human operations are expensive.

Because of this, there is an implicit assumption that the use of human processing power in such systems will be judicious. After all, there is a point at which human "processors" will simply refuse to perform any more computation. Much effort has been put into learning how to best incentivize human processors to perform computation (Mason and Watts, 2010; Singer and Mittal, 2011). *Games with a Purpose* try to make participation more entertaining for the human (von Ahn, 2006), thereby increasing their willingness to contribute. However, to date few mechanisms have been developed for comparing the algorithmic processes underlying human computation systems independent of the details of their implementation. While questions of recruiting and incentivizing human contributors are by no means unimportant, in this work we specifically investigate the complexity of the underlying processes at the algorithmic level independent of these factors.

3. ADAPTING THE HUMAN ORACLE MODEL FOR HUMAN COMPUTATION

Throughout the remainder of this paper, we will adopt two slight relaxations of the assumptions about the behavior of the Human Oracle as differentiated from traditional set-theoretic definitions of an Oracle. These relaxations are meant to enable meaningful comparison between simple *Games with a Purpose*-style human computation system by capturing a small part of the variability inherent in leveraging humans in these computational processes, such as fuzzy interpretations of "correct" responses and the existence of short term memory. While we do not suggest that the following model is complete, we hope that it will provide a toehold for future exploration in this important area.

3.1. Variability in Human Oracle Responses

Under the standard set-theoretic definition of an Oracle, any two Oracles to the same problem are equivalent with respect to the answers they return to a given query. In contrast, one could reasonably expect that different Human Oracles to the same problem may return different answers when queried on the same input, if more than one appropriate answer exists. Whenever there is potential for ambiguity in processing stimuli, there may be more than one valid response for any given input. However, a given individual may strongly favor one response over another.

We characterize this behavior as follows. Under this model, we will assume that there exist **finitely many** reasonable responses for any query/input pairing:

$$R_Q(x) = \{r_1, r_2, \ldots, r_{n-1}, r_n\}$$

where r_i is a reasonable response to query Q on input x. We then state that any valid Human Oracle always returns one such reasonable answer, but that we can't predict which one they may decide to return. We can express this behavior by defining the Human Oracle H as having a probability distribution over the collection $R_Q(x)$:

$$D_{H(Q,x)} = \{\langle r, P_H(r_i)\rangle | r_i \in R_Q(x), 0 \leq r_i \leq 1\}$$

where $P_H(r_i)$ is the probability that Human Oracle H returns response r_i when passed query Q on input x, and $\sum_{i=1}^{n} P_H(r_i) = 1$.

In the simplest case, $n = P_H(r_n) = 1$. That is, if there is only one reasonable response, the Human Oracle will return that response with probability 1. When there are multiple reasonable responses, the Human Oracle's probability distribution may heavily favor some subset of responses. We suggest that this probabilistic behavior helps capture the influence of individual differences inherent in any human population. These inconsistencies may be due to different lived experiences, internal biases, or preferences. In addition to individual differences, this distribution may be influenced through incentivization. This may happen *a priori*, such as in systems that incentivize the generation of short responses over more verbose ones, or the distribution may be changed *on-the-fly*, such as in gameified systems where the players may be asked to match (or avoid matching) a partner's responses.

For simplicity, we will exclude nonterminating algorithmic executions from consideration under this model by assuming that the intersection between any two Human Oracle's response sequences will be nonempty, provided they are queried sufficiently many times on the same input. This helps capture the notion of *collective intelligence* relied upon in many human computation applications, and enables meaningful comparisons between the worst-case performance of various approaches. When such an assumption is not made, the worst case performance of any algorithm that utilizes output agreement is trivially infinite; by repeatedly matching pairs of Human Oracles with non-overlapping response sets, the system could continue to query without ever being able to validate an answer.

In practice, individual differences sometimes dictate that the probability of intersection between two people's response sequences is arbitrarily small. For example, one partner may be unfamiliar with an animal not native to their region or a landmark they have never seen, and may therefore not describe it using the same terminology as their more familiar collaborator. To deal with this reality, many real-world systems implement a timeout mechanism to ensure that an algorithm does not spend too much time on a potentially fruitless pairing of Human Oracles. While beyond the scope of this paper, it is also interesting to consider the effects of relaxing the nonempty intersection assumption on algorithmic analysis and expected performance when given some additional information about the distribution of knowledge in the sample population.

3.2. Persistence of Previous Responses

If the same Human Oracle is queried more than once on the same input during the execution of an algorithm, we may wish to assume that it will be aware of its previous responses and will only return each answer once. This is akin to assuming that upon reading the input, the Human Oracle constructs a predefined sequence of answers by ordering their possible responses in decreasing order of probability:

$$A_{H(Q,x)} = (a_1, a_2, \ldots, a_n)$$

where:

$$P(a_{i+1}) < P(a_i) \ \forall\ 1 \leq i \leq n$$

The Human Oracle will answer each query about that particular input by simply reporting the next unused element in the sequence. This reflects human short-term memory, and can be simulated by recording previous responses in the main algorithm and passing the entire history back as part of the input to a non-persistent Oracle.

3.3. Additional Assumptions

Additionally, we presume that the Human Oracle can efficiently generate an answer to the queries we pose. In traditional computational models, it is assumed that the Oracle can interpret and respond correctly to the query in constant time. However, it is also acceptable to consider Oracles with other (bounded) response time complexities. With Human Oracles, we do not necessarily know how long it takes a person to solve the problem. For simplicity, we will assume a constant cost for each query to the Human Oracle given a particular affordance, which enables us to consider the complexity of two algorithms leveraging the same kind of Human Oracle in terms of the number of queries required.

Finally, the study of human computation presumes the existence of problems for which humans are *faster, more accurate, more robust, or otherwise superior to* any known machine algorithm. To that end, we only consider problems in which the Human Oracle's answers are integral to computing the solution. That is, the algorithm querying the Human Oracle cannot efficiently generate answers to its own queries, and must rely on (and potentially validate) the responses it receives from the Human Oracle.

We believe that these adaptations result in a model that more closely resembles observed behavior in systems involving human computation, and help capture some of the ambiguity inherent in many interesting human computation problems. In the following section, we use this model as a lens to explore various problems that fall under the umbrella of image labeling. We do not mean to imply that these are necessarily "canonical" or "complete" problems for human computation, as this concept is yet ill-defined. However, we believe that a close examination of well-studied problems through this lens may provide insight into the structure of human computation algorithms. We hope that this will serve as an initial benchmark by which other problems may be measured as we continue to explore the space of human computation.

4. IMAGE LABELING UNDER THE HUMAN ORACLE MODEL

In this section, we explore how the Human Oracle Model can be used to describe the underlying algorithmic behavior and relative performance of human computation systems. The Image Labeling examples considered herein are canonical exemplars in the field; they are by no means a comprehensive sampling of human computation in the wild. Instead, we leverage these simple examples and narrow use case in order to provide a concise vignette into the kinds of low-level evaluations and algorithmic comparisons that are made possible through the adoption of abstract models like the Human Oracle Model. In some cases, we have elected to model a slight variation of a system in the interest of facilitating a more interesting comparison. When this is the case, we will clearly document any modifications and provide justification for the change.

4.1. The ESP Game

The ESP Game[2] (von Ahn and Dabbish, 2004) is a human computation system designed to produce validated labels for images on the web. Each image is displayed to a randomly-assigned pair of human collaborators, who are then asked to label the image in a finite amount of time. Because the human players cannot communicate with one another as they try to "agree" by guessing the same label, the dynamics of the game incentivize them to try guessing short, intuitive labels. A label is accepted once some number of pairs have agreed on it, and is then added to a list of `TABOO` words for that image. Future pairs are presented with the `TABOO` words in addition to the image, and these words are not accepted if guessed. This encourages the generation of new labels by future pairs, rather than simply repeating previously validated labels. For the purposes of this analysis, we will assume that just one pair must agree for a label to be accepted and that the computation continues until a match is found. The resulting output is a new description of the image that has been validated by both Human Oracles.

Recall that each Human Oracle has a finite list of labels they could use to describe a given image. In the best case, the algorithm terminates after only 2 queries, one to each Human Oracle whose first choice labels are a match. In the worst case, the Human Oracles' response lists A_{H_1} and A_{H_2} are exactly inverted and both have nonzero probabilities for all n possible labels. If this is the case, then each of the first n queries ($\frac{n}{2}$ to each Human Oracle) would result in a unique label before the $(n+1)^{st}$ query in forces a match. In the event that either H_1 or H_2 cannot return a new label, we assume that the computation will throw an error. When this occurs, the assumption that all pairs have a nonempty intersection in their response sequences allows us to infer that all valid labels for the input image are already listed in the `TABOO` list; if this were not the case, then the Human Oracles would have guessed the missing label.

4.2. KissKissBan

KissKissBan (Ho et al., 2009) is another human computation system designed to produce short, validated labels for images on the web. The authors note that "players tend to give easier and more generic descriptions [when playing the ESP Game], and therefore the diversity of the output becomes limited (Ho et al., 2009)." This system suggests an extension of the ESP Game intended to generate more creative labels by "introducing competitive element into the game (Ho et al., 2009)." Each image is shown to three online players. Two players are collaborating, as in the ESP Game, to try to guess the same label for the image. The other player, the Blocker, attempts to block the collaborative pair from matching by guessing a set of obvious labels at the onset of each

[2]Because of the widespread success of the original ESP Game and its many subsequent variants, many refer to the general collection of similarly incentivized input-agreement human computation systems as *ESP Games*. Because we will be exploring the relationships between several members of this larger class of systems, in this publication we will reserve the term "the ESP Game" to refer only to the original 2-player image labeling game published by von Ahn and Dabbish.

round. These blocked labels are hidden from the collaborators, who are incentivized to match on a non-blocked word before their time runs out and are penalized for guessing blocked words. If they fail to match on a new word, the Blocker wins.

There are three ways a label can be validated during the game: (1) H_1's label matches one from $H_{Blocker}$, (2) H_2's label matches one from $H_{Blocker}$, or (3) H_1 and H_2 match on a label as in the ESP Game. Note that while matching on a blocked word produces a validated label, the game ends only on a match between the two collaborators. The resulting output is a *set* of labels that have each been validated by at least two Human Oracles.

We will presume that the Blocker generates $k-1$ labels at the onset of each game. In the minimal case, H_1 and H_2 match on their first label and this label is not blocked, requiring a total of $k+1$ queries to generate a single label. Unlike with the ESP Game, the minimal case is not optimal in terms of minimizing queries-per-label. In the best case, the responses of H_1 and H_2 differ on first $k-1$ queries, but each response matches one of the $k-1$ blocked labels. They then match on their next guesses, requiring a total of $2k$ queries to generate k labels. In the worst case, the Blocker then responds to $k-1$ queries to generate the Blocked list, but none of these are matched by the collaborators. H_1 and H_2 are exactly inverted on their ordering of the remaining $(n-(k-1))$ responses, and the next query forces a match. When this is the case, KissKissBan requires $(k-1)+(n-(k-1))+1 = n+1$ queries to generate a single label.

4.3. Comparing the ESP Game and KissKissBan

Intuitively, the original implementation of the ESP Game and KissKissBan appear very similar both in terms of the problem they are trying to solve as well as the approach to finding a solution. Because their underlying problems are equivalent and their Human Oracles differ in number but not in function, we can directly compare the performance of the ESP Game and KissKissBan algorithms under the Human Oracle Model. We will begin by demonstrating that the worst case performance of the ESP Game requires no more queries per label than the worst case performance of KissKissBan.

Proof: Recall that in the worst case, KissKissBan returns just a single label with a large number of queries to the Human Oracles. All $k-1$ queries to $H_{Blocker}$ were wasted because none of the `BLOCKED` labels were matched, and the collaborators go $\frac{n-(k-1)}{2}$ rounds before finding a match for a total cost of $n+1$ queries. In this case, returning a single label could have been accomplished using one round of the ESP Game at an identical cost of $n+1$ queries. While the two Human Oracles may take just as long to find a match, there is no added benefit to including a third Human Oracle in the worst case. Thus, the worst-case number of queries to generate a single label in KissKissBan is equal to the worst-case cost of the ESP Game. ■

We will next demonstrate that the best case performance of KissKissBan requires no fewer queries per label than the best case performance of the ESP Game.

Proof: In the best case, KissKissBan returns k unique labels using $2k$ queries to the Human Oracles: $(k-1)$ to $H_{Blocker}$ to set up the `BLOCKED` list, $(k-1)$ queries divided between H_1 and H_2, each of which matches a unique word on the `BLOCKED` list, and 2 final queries, one to each of H_1 and H_2, on which they match. This match causes the algorithm to terminate with a cost-per-label of 2. In the best case performance of the ESP Game, the pair is able to match on their first try for a total of 2 queries to the Human Oracles to generate a single label. Thus, the minimum number of queries per label in the best-case performance of KissKissBan is equal to the best case cost of k rounds of the ESP Game. ■

From an algorithmic perspective, KissKissBan demonstrates no advantage over the ESP Game in terms of the **number of queries per label**. However, it is reasonable to argue (as do the authors) that KissKissBan (Ho et al., 2009) may produce "less obvious" labels than the ESP game in the short term. That is, the game dynamics in KissKissBan encourage the non-Blocking players to de-prioritize labels that they believe are likely to be guessed by the Blocker. More specifically, they are encouraged to avoid their initial top choices, assuming that those guesses will also be the top choices of their opponent. Despite this reordering, if played for enough rounds this model suggests there are no labels that KissKissBan would produce that would not also eventually be discovered in the ESP Game. Because both games are played using exactly the same kind of Human Oracle, the pool of potential labels is the same in both games; they may just be validated in a different order. This suggests that the difference in initial labels is due more to the incentive structure of the game than to any underlying computational differences, and challenges the claim that either algorithm provides *better* labels – they provide the same labels, in a different order. This is an important distinction: it enables us to isolate the cause of the effects we observe in the wild, and begin to explain the phenomenon in in computational terms. This comparison validates that the Human Oracle Model enables quantifiable comparison between algorithms that leverage the same affordance to solve the same problem.

4.4. **Peekaboom**

Peekaboom (von Ahn et al., 2006) is a human computation system designed to augment image labels with information about the location of the objects being described. Two players are partnered at random and are each assigned a role: Peek and Boom. Boom is presented with an image and a word, and Peek is presented with a blank screen. Boom is tasked with revealing just enough of the image to Peek so that she can guess the word. As Boom clicks on parts of the image, a small region of the image under the clicked location is revealed, and the incomplete image is sent to Peek.

The task given to Peek is identical to players of both the ESP Game and KissKissBan: given an image (in this case, an incomplete image), provide a description. Both players are incentivized to reveal and guess as efficiently as possible. The game infers that if Peek is able to guess the word, then Boom must have revealed the correct location. Once Peek has successfully matched the original word, a minimal bounding box is computed from the regions revealed by Boom. Experimental data suggest that the bounding boxes produced by multiple pairs when averaged tend toward minimally bounding the region containing the object. In this problem, we either assume that the textual description has been validated a priori, or that the computation will throw an error if the object does not appear in the image.

In the best case, one reveal from Boom is sufficient for Peek to guess correctly on the first try, for a total of 2 queries to validate the label. In the worst case, Boom must reveal all subregions of the entire $m \times m$ image before Peek can identify the correct label, resulting in a total of $O(m^2)$

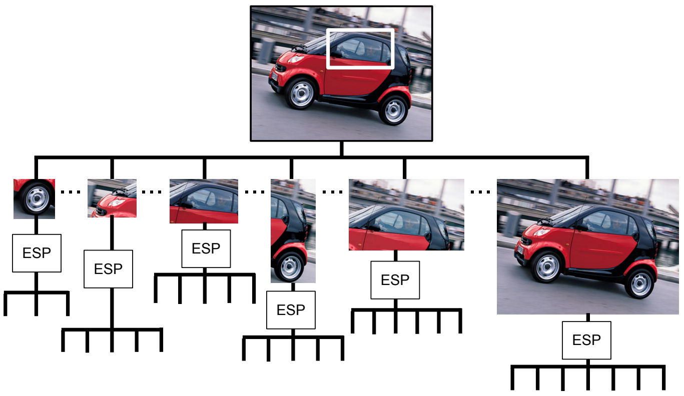

Figure 1.* *Bounding an object labeled* driver *using iterative queries to the ESP Game on subimages.

queries to validate the label. In contrast to the two previous applications, in which humans are asked perform the same task and their responses are used to to verify one another, Peekaboom uses implicit validation on humans performing different tasks. This hints at a difference in the underlying computational machinery, which will be further discussed later in this work.

4.5. Comparing the ESP Game and Peekaboom

While both the ESP Game and Peekaboom compute on a single image and ask a a Human Oracle to describe the image as part of their computation, their underlying problems appear different. However, we can use reduction to demonstrate that the ESP Game can be used to solve the problem being solved in Peekaboom (bounding an object in an image).

Proof: Given an image I and a label describing an object to bound, nondeterministically select a subimage I'. On I', iterate the ESP Game to return all possible descriptions. If any of the returned labels matches the input label, return the boundary of the subimage as the bounding box. ∎

The number of possible subimages is limited by the size of the image. As before, the number of possible valid descriptions for any input image is also finite due to the limitations of both image resolution and language. Thus, if the label is valid for the subimage, it will eventually show up as one of the suggested descriptions returned by the ESP Game. We are therefore guaranteed that this nondeteriministic “guess-and-check” method will eventually yield a correct bounding box. This reduction is depicted graphically in Fig. 1.

We now demonstrate that the maximum number of queries to the Human Oracle using either Peekaboom or repeated rounds of the ESP Game required to bound an image are both polynomially bounded in the size of the image, these polynomial bounds are not equivalent.

Proof: Recall that the goal of Peekaboom is to return a minimal $w \times h$ bounding box surrounding the described object in the image, and that this is accomplished by having Boom sequentially reveal parts of the image to Peek. Assume without loss of generality that the size of the image is $m \times m$, and that the size of each revealed region is $r \times r$, where $0 < r < m$. The smallest possible bounding box, where $w = h = r$, would be returned in the case that Peek was able to guess the word after a single reveal. In the worst case $w = h = m$, because Peek may not be able to guess the word before seeing the entire image, which could require at most $2 * (\frac{m}{r})^2 = O(m^2)$ queries to the Human Oracles.

As indicated above, we can repeatedly invoke the Oracle-ESP algorithm on each subimage in ascending order of size until either the algorithm returns a matching label or we exhaust the possible labels for the subimage without finding a match, and move on to the next one. Because (1) the label given as input to Peekaboom was validated a priori, (2) there are finitely many valid labels for any image, and (3) the Oracle-ESP algorithm will eventually return all valid labels, we can be assured that this process eventually terminates. Because we evaluate subimages in increasing order of size, this process guarantees that the first subimage on which we find a match is minimal.

The total number of times we must play the ESP Game is determined by the number of subimages that must be evaluated. The smallest bounding box that could be returned by Peekaboom is the size of one revealed region, and so we need only evaluate subimages that are at least $r \times r$, and that are at most the entire image. The number of possible subimages ranging from size $r \times r$ to $m \times m$ is:

$$\sum_{w=r}^{m} \sum_{h=r}^{m} (m-w+1)(m-h+1) = O(m^4)$$

thus requiring on the order of $O(m^4)$ queries to the Human Oracles across all executions of the algorithm. Thus, the worst-case number of queries needed to bound an object using only the ESP Game grows asymptotically faster than the number of queries needed using Peekaboom. ■

In the proofs above, we used brute force to illustrate the relationship between the ESP Game and Peekaboom. This demonstrates the relationship between *labeling an image* and *locating an object in an image* in an intuitive manner, but we reiterate that this is not intended as a prescription for future design. In practice, because this method requires an exhaustive search, this approach would not be an effective use of human computation resources. The average case performance could likely be significantly improved by making more intelligent decisions about which subimages to send to the subroutine for validation. We could, for example, start with a well-distributed subset of subimages of each size. This has the potential to greatly reduce the number of calls to the subroutine because it could de-prioritize redundant rounds on uninteresting regions of the image without sacrificing accuracy. We could also select regions according to content by preprocessing using an image segmentation algorithm. This would increase the amount of machine computation, in exchange for a reduction in the number of queries to the Human Oracles. However, these heuristics would not alter the underlying differences between these algorithms.

5. IMAGE VS. AUDIO LABELING

In the above comparisons, we demonstrated that the ESP Game can be used to approximate several other human computation algorithms. However, this is not meant to suggest that the ESP Game is somehow universal or complete. For example, consider whether a successful image labeling algorithm such as the ESP Game (von Ahn and Dabbish, 2004) can be used to label other stimuli such as audio. Such a reapplication was attempted by the original designers to create a system called TagATune (Law et al., 2007). Despite being identical to the ESP Game in nearly every way with the exception of the human affordance, this first iteration failed miserably; people simply couldn't agree on labels for most of the input. In a second iteration, the designers found that asking the users to decide whether or not they thought they were listening to the same thing was far more successful for audio labeling than explicit matching (Law and von Ahn, 2009), although this introduces significantly more noise into the resulting dataset.

This would indicate that though the human is superficially being asked to perform a similar task, the underlying information processing is fundamentally different for images versus audio. Our lived experience might lead us to speculate that the human might be sampling their responses from a much larger search space. That is, perhaps the sets $R_Q(x)$ from which Human Oracles are drawing their responses are dramatically larger for audio inputs than for images. This makes intuitive sense; after all, audio lacks the same tangible, concrete concepts like *chair* and *grass* upon which we often anchor our labels for images. In addition, one might suggest that the fact that the input is continuous rather than discrete might play some role. Whatever the underlying reason, this suggests that in some sense audio labeling is a *harder problem* than image labeling, and that it therefore requires more powerful machinery to solve. In the following section, we define several dimensions which can be used to compare the relative strength of systems leveraging human computation and demonstrate that these dimensions define preliminary complexity classes on the space of human computation.

6. DIMENSIONS AND COMPLEXITY CLASSES

As demonstrated in the previous section, the number of required operations is one intuitive metric by which we may order a collection of algorithms for solving a problem. Indeed, this is analogous to traditional notions of computational work. Because we lack a mechanism for converting between units of *human work* and units of *machine work*, the $\langle \Phi_H, \Phi_M \rangle$ notation introduced by Shahaf and Amir (Shahaf and Amir, 2007) can prove useful.

Recall for example the techniques discussed previously for identifying the location of an object in an $m \times m$ image. Using Peekaboom, the number of queries to the Human Oracles is bounded by $O(m^2)$. The cost to the machine is a simple comparison between each of Peek's guesses and the input label, and so $\Phi_M = \Phi_H = O(m^2)$ as well. In the second approach using the ESP Game as a subroutine, the number of queries to the Human Oracles could be as high as $O(n^2 * m^4)$ in the event that all n labels need to be validated for each subimage before we arrive at the input label. The machine must then compare the value returned by each query to the collection of previous guesses for that round to determine if there has been a match. Assuming an efficient data structure, each lookup or insertion would incur a cost on the order of $\log(n)$. In addition, each of the n possible returned labels must be compared against the input label. This results in a total machine cost of $O(n^2 * m^4 * \log(n) + n)$. Comparing these two tuples, it is clear that Peekaboom is a more efficient method for bounding an object in an image than the ESP Game approximation in terms of both human and machine computation.

Perhaps more interestingly, the examples given demonstrate that an algorithm requiring **more information** (Peekaboom requires a predefined label) as well as **interactivity** (Boom must be able to touch the image) can be simulated using a polynomial number of calls to an algorithm with limited interactivity and unrestricted search space (ESP). This sheds important light on the *value* of this additional information and power, and the scale of the cost incurred if we are forced to solve the same problem using an intuitively "weaker" machine.

This notion of "stronger" and "weaker" suggests that the way in which human computation is leveraged as part of an algorithmic process may be used to induce *complexity classes* that partition the space of human computation. By developing an intuition about the ways in which problems group together in terms of how their solutions leverage human computation as a resource, we can better equip ourselves to exploit what we already know to solve novel problems. In the following sections, we will discuss several additional dimensions along which human computation may be classified and compared.

6.1. Query Order

First, we can consider whether or not the sequence of queries to the Human Oracle can be determined in advance. In an algorithm with **predetermined query order**, we claim that there exists some function:

$$f : I \rightarrow (q_1, \ldots, q_n)$$

that takes as input a problem instance I and generates a finite sequence of queries $(q_1, \ldots, q_n)$ that will be passed in order to the Human Oracle. Because the sequence can be generated *a priori*, it follows that the position of any query q_i must not depend on the Human Oracle's previous answers. In these systems, the Human Oracle cannot influence the order in which it receives and responds to specific queries. A general example of a process that uses predetermined queries is semi-supervised machine learning. In these techniques, the Human Oracle is asked to label a set of training data which is then used to infer a classification function. While the resulting classification function is dependent on how the training data is labeled, the points to be used as training data are determined in advance.

Relaxing this restriction yields algorithms whose future queries may be contingent on the Human Oracle's answers to previous queries. In an algorithm with **adaptive query order**, we claim that there exists some function:

$$\begin{aligned} f : \{I, \emptyset\} &\rightarrow q_1 \\ f : \{I, (a_1)\} &\rightarrow q_2 \\ &\vdots \\ f : \{I, (a_1, \ldots, a_n)\} &\rightarrow q_{n+1} \end{aligned}$$

that takes as input a problem instance I as well as $(a_1, \ldots, a_n)$, the sequence of responses from the Human Oracle so far, and generates the next query (q_{n+1}). An excellent example of adaptive querying is *active learning*. In active learning algorithms, the Human Oracle is first asked to label some small set of training data. Based on their responses, the algorithm reclusters the data. It then

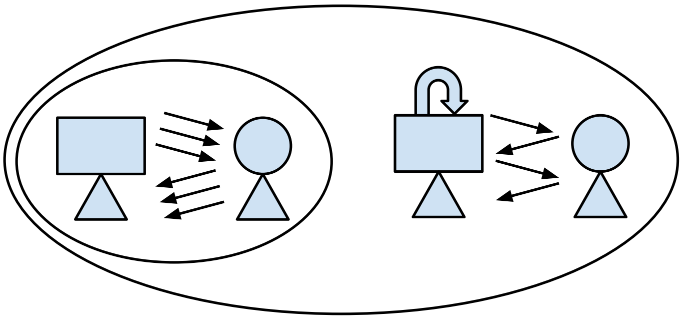

Figure 2. ***Illustration of the relationship between adaptive and predetermined query order machines. In a predetermined query order, all queries to the Human Oracle must be determined*** **a priori;*****. In an adaptive query order, subsequent queries to the Human Oracle may be adapted based on the response to previous queries. Any result that can be computed on a machine with predetermined query order (left) can also be computed on a machine with adaptive query order (right).***

selects a new subset of points about which it is least confident and submits these to the Human Oracle to label. The selection of each subsequent collection of points is directly related of the labels provided in the previous round. This process continues iteratively until some confidence threshold is met.

Any result that can be computed on a machine with predetermined query order can also be computed on a machine with adaptive query order; the adaptation would simply be to do nothing with the additional information. The inverse is not true, as the decisions made by an adaptive query machine regarding which question to ask next may not be predeterminable. Thus, the class of problems solvable using a predetermined query machine is contained within the class of problems solvable using an adaptive query machine (Fig. 2).

6.2. **Oracle Responses**

Mirroring the previous dimension, some systems may treat the Human Oracle's responses to queries as largely **independent** of one another. One example of a real-world system where this is true is reCAPTCHA (von Ahn et al., 2008), a human computation system for optical character recognition (OCR). If the same human is asked to pass several reCAPTCHA instances, any response bias due to priming or learning would likely be negligible. Thus, for analytical purposes, we can presume that each of her responses is independent. In practice, processes leveraging such sequence-independent responses may be parallelizable or perhaps more robust to interruption or participant turnover. In such cases, there would be no discernible difference between asking 10 Human Oracles to each answer 10 queries, or asking 100 different Human Oracles 1 query each. Because these Human Oracles have no practical awareness of their previous responses, we refer to them as *forgetful*.

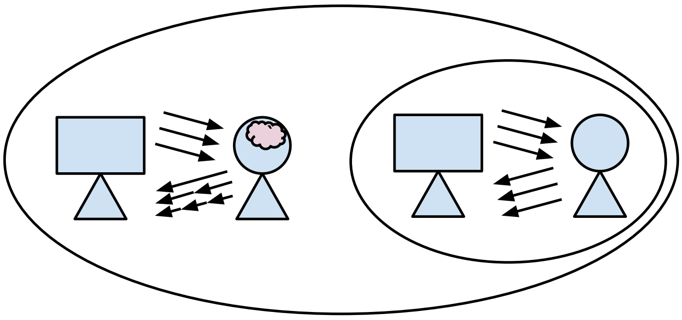

Figure 3. ***Illustration of the relationship between machines that use wise and forgetful Human Oracles. A forgetful Human Oracle does not gain any information from answering previous queries; a wise Human Oracle may use previous queries to influence its responses in the future. Any result that can be computed using a forgetful Human Oracle (right) can also be computed using a wise Human Oracle (left).***

It is sometimes useful to endow the Human Oracle with some amount of persistent memory regarding the query sequence. In these systems, the Human Oracle may be able to **modify its future behavior based on previous events**. In the simplest sense, this could be used to ensure that the Human Oracle does not return the same answer twice as in the previous examples. In other scenarios, we may wish to enable computation on this sequence in order to model more nuanced adaptations, such as learning or fatigue. For example, complex systems such as visual analytics tools require that the Human Oracle be able to learn from a prior sequence of actions and responses and subsequently modify its future behavior. Note that while we continue to develop more robust models of human memory and its limits, we may abstract the specifics of *how* the Human Oracle remembers and instead include the cost of accessing this memory in the query cost. Because these Human Oracles can remember and compute on their previous answers, effectively gaining knowledge from their response history, we refer to them as *wise*.

As with the previous dimension, any result that can be computed on a machine with a forgetful Human Oracle can also be computed on a machine with a wise Human Oracle; when simulating a forgetful machine, the wise Human Oracle's memory remains unused. The inverse is not true, as the responses returned by a wise Human Oracle may not be parallelizable. Thus, the class of problems solvable using a forgetful Human Oracle is contained within the class of problems solvable using a wise Human Oracle (Fig. 3).

6.3. Supplemental Information

We can also consider whether or not the Human Oracle can inject supplemental information into the computation process on its own. This is tantamount to asking whether or not the Human Oracle is allowed to provide answers to questions that were not directly asked. In many existing human computation algorithms, the human or collection of humans is asked to perform a *specific* computational process as a subroutine to a larger algorithm, such as labeling a particular image.

Under this restriction, the Human Oracle does not have any power to interject new information or redirect the computational process. This is consistent with the standard Oracle Turing Machine model; the Oracle is simply a set, and can only answer questions about membership in that set. The algorithms under consideration in this work and in many other *Games with a Purpose*-style systems, such as Fold.it (Cooper et al., 2010) for finding protein foldings, fall within this category.

Alternatively, the Human Oracle may be given some autonomy regarding the interjection of supplemental information into the computational process. That is, in addition to responding to queries generated by the computational process, the Human Oracle may spontaneously provide new information that may then impact future computation. This adds an additional level of computational power to the Human Oracle. For example, the Human Oracle may have its own computational agenda such as in the use of Visual Analytics systems, and may therefore inject supplemental information to direct the computation in a specific direction. Indeed, we suggest that this supplemental information may in fact be integral to the success of many Visual Analytics systems that leverage the domain expertise or lived experience that is yet impossible to simulate within purely mechanical analysis.

7. COMPARING HUMAN COMPUTATION SYSTEMS IN COMPLEXITY SPACE

In contrast to previous schema for comparing human computation systems which rely on nominal classification dimensions (Quinn and Bederson, 2011), each of the dimensions introduced here has an implicit notion of magnitude that induces an partial ordering on different algorithms and problems. Categorizing along these three dimensions, many of our intuitions about the relative strength of existing techniques are captured. For example, the algorithm underlying reCAPTCHA can be computed using a less powerful Human Oracle than the ESP Game. In reCAPTCHA, the human is simply a visual decoder, and the machine takes care of compiling and aggregating the results. In the ESP Game, more responsibility is placed on each individual human to avoid performing redundant computation in order to generate a selection of unique labels. Similarly, we see that active learning requires a more powerful use of human computation than semi-supervised learning. We presume that by enabling more careful selection of user constraints or labels, the set of datasets that can be classified using active learning is broader than the set of datasets that could be classified using semi-supervised learning, given the same amount of supervision.

Because they define a partial ordering between problems, these dimensions can be used to establish preliminary *complexity classes* within the space of human computation (Fig. 4). We suggest that these classes are complementary to those in traditional computational complexity. Indeed, we may consider these hierarchies to be orthogonal. That is, there could exist human-computer collaborative systems with all combinations of $\langle \Phi_H, \Phi_M \rangle$. For example, the ESP Game lives at the intersection of *predetermined query order, wise Human Oracle* and $O(n^2) \in \mathtt{P}$, while Fold.it exists where *adaptive query order, wise Human Oracle* meets $\mathtt{NP}$.

This hierarchy is still a work in progress; many of the relationships between these classes are still unknown (Fig. 5). For example, how does the class of problems solvable by a *fixed query order, forgetful Human Oracle* machine with *supplemental information* relate to the class of problems solvable using a machine with *adaptive query order, a forgetful Human Oracle* and *no supplemental information*? How does this compare to a machine with *fixed query order, a wise Human*

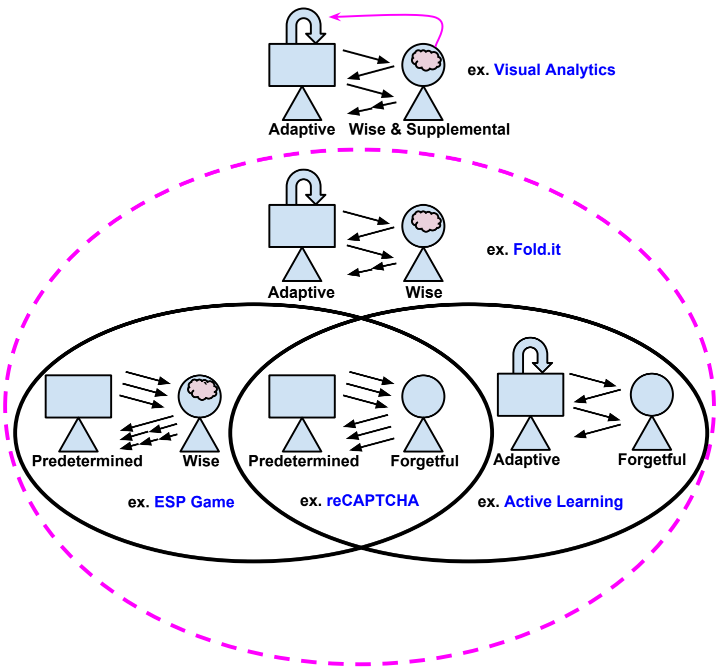

***Figure 4.** Preliminary hierarchy of complexity classes in human computation defined under the Human Oracle Model.*

Oracle and *no supplemental information*? These questions, along with the discovery of canonical or **complete** problems in this space, are the subject of our future research in this area.

8. DISCUSSION

The Human Oracle Model provides a critical first step in quantifying human computation, and helps us to better understand the intricate relationships among different problems and problem families when viewed through the lens of human computation. That said, this work only just scratches the surface of this potentially rich area for future research. To start, this paper considers only boundary cases: that is, what are the upper and lower bounds on the work that is being done? While it is beyond the scope of this initial exercise, it would also be very informative to assess the average case performance of these algorithms as well.

In addition, this model ignores some very real factors present in any system involving the variability of biological computation. In the following sections, we discuss some of the limitations of this model, as well as motivate continued research in this area.

8.1. Imperfect Oracles

Under this model, there is an explicit assumption the Human Oracle will always be able to provide the correct answer at a fixed cost. In reality, humans don't work this way. Intuition and experience indicate that humans eventually get tired or bored, and as a consequence their speed and accuracy suffer. In addition, individual differences in ability and cost are absent. In the real world, not all humans are equal in their capacity to answer the questions we ask. Some are more skilled or have better training, and their expertise comes (we presume) at a higher cost.

Similar issues have arisen in the area of active learning, which has historically assumed a single tireless, flawless, benevolent Oracle was always available to provide labels for its training data. *Proactive learning* relaxes these assumptions, adopting a decision-theoretic approach to match one of a collection of (possibly imperfect) Oracles to each instance (Donmez and Carbonell, 2008). More recent work in the area of *multiple expert active learning* (MEAL) improves upon this model by incorporating load balancing to ensure that no worker has to shoulder an inequitable share of the burden (Wallace et al., 2011). These methods assume there exists some method to model both how hard any single problem instance is, as well as how costly and effective a given worker is.

8.2. Quantifying the Human Brain

This highlights another problem: as of this writing, there does not exist any reliable method for quantifying how hard a human has to work in order to accomplish a given task. Because we don't fully understand the fundamental operations of the human brain or how they assemble to perform computation, it is not yet possible to calculate a precise per-operation cost. As such, at present this model cannot actually tell us *how much work* the human is doing; it only tells us how many times the human is working (query complexity). When the task is comparable, such as when comparing various image labeling algorithms, this does not pose a significant problem. But what about comparing dissimilar processes?

While cognitive modeling techniques can help us to understand the interplay between stimulus and response, existing architectures are not designed to determine the "complexity" of the model itself. Unobtrusive brain sensing methods are currently under development and have shown promise in detecting task difficulty (Girouard et al., 2009), but the information revealed is not yet refined

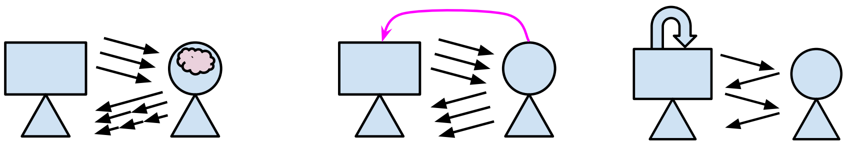

Figure 5. Open question: how does a **fixed query order, forgetful Human Oracle** *machine with* **supplemental information** *(center) relate to the class of problems solvable using a* **fixed query order, wise Human Oracle** *machine with* **supplemental information** *(left) or an* **adaptive query order, forgetful Human Oracle** *machine with* **no supplemental information** *(right)?*

to a per-operation granularity. Thus, from a cognitive science perspective, there is presently no mechanism for quantifying the computation performed by the human brain. In order to form a complete model of human computation, it is critical that we continue to develop more nuanced models of the human brain and to incorporate these models into the evaluation of algorithmic complexity and performance in human-machine collaborative systems.

9. CONCLUSION

The importance of understanding human computation as part of a larger computational complexity system is not limited to improving algorithm design. Augmenting computational complexity models to incorporate human computation can expand our understanding of what can be computed, as did the development of probabilistic and parallel computation. The development of complexity measures for human computation may play a significant role in the broader adoption of human computational methods. Robust models of *how humans fit* into the grand scheme of computational tools is essential to promoting wider understanding of human effort as a legitimate and measurable computational resource.

In this work, we introduced the Human Oracle Model as a method for characterizing and quantifying the *use of human processing power as part of an algorithmic process*. We demonstrated the utility of this model for comparing and analyzing several well-known human computation systems for image labeling and described how this model can be used to characterize the space of human computation. In closing, we discussed the model's limitations and its potential for broader impact. Through this research, we hope to form a more holistic picture of the interrelationship between human and machine computation, and to develop a robust theoretical model for the analysis of systems involving their collaboration.

10. REFERENCES

Arora, S and Barak, B. (2009). *Computational complexity: a modern approach.* Cambridge University Press.

Bertini, E and Lalanne, D. (2010). Investigating and reflecting on the integration of automatic data analysis and visualization in knowledge discovery. *SIGKDD Explorations Newsletter* 11, 2 (2010), 9–18.

Cooper, S, Khatib, F, Treuille, A, Barbero, J, Lee, J, Beenen, M, Leaver-Fay, A, Baker, D, Popovic, Z, and others, . (2010). Predicting protein structures with a multiplayer online game. *Nature* 466, 7307 (2010), 756–760.

Crouser, R and Chang, R. (2012). An Affordance-Based Framework for Human Computation and Human-Computer Collaboration. *Visualization and Computer Graphics, IEEE Trans. on* 18, 12 (2012), 2859–2868.

Donmez, P and Carbonell, J. (2008). Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proc. 17th ACM Conf. on Information and knowledge management*. ACM, 619–628.

Girouard, A, Solovey, E, Hirshfield, L, Chauncey, K, Sassaroli, A, Fantini, S, and Jacob, R. (2009). Distinguishing difficulty levels with non-invasive brain activity measurements. In *Human-Computer Interaction*. Springer, 440–452.

Ho, C, Chang, T, Lee, J, Hsu, J, and Chen, K. (2009). KissKissBan: a competitive human computation game for image annotation. In *Proc. SIGKDD Workshop on Human Computation*. ACM, 11–14.

Law, E and von Ahn, L. (2009). Input-agreement: a new mechanism for collecting data using human computation games. In *Proc. 27th SIGCHI Conf. on Human factors in computing systems*. 1197–1206.

Law, E, von Ahn, L, Dannenberg, R, and Crawford, M. (2007). Tagatune: A game for music and sound annotation. *Proc. of ISMIR (Vienna, Austria)* (2007).

Lawler, E. L, Lenstra, J. K, Kan, A. R, and Shmoys, D. B. (1985). *The traveling salesman problem: a guided tour of combinatorial optimization*. Vol. 3. Wiley New York.

MacGregor, J. N and Ormerod, T. (1996). Human performance on the traveling salesman problem. *Perception & Psychophysics* 58, 4 (1996), 527–539.

Mason, W and Watts, D. (2010). Financial incentives and the performance of crowds. *ACM SigKDD Explorations Newsletter* 11, 2 (2010), 100–108.

Quinn, A and Bederson, B. (2011). Human computation: a survey and taxonomy of a growing field. In *Proc. 29th SIGCHI Conf. on Human factors in computing systems*. ACM, 1403–1412.

Shahaf, D and Amir, E. (2007). Towards a theory of AI completeness.. In *Logical Formalizations of Commonsense Reasoning*. 150–155.

Singer, Y and Mittal, M. (2011). Pricing Tasks in Online Labor Markets.. In *Proc. Workshop on Human Computation at the 25th AAAI Conf. on AI*.

Turing, A. (1938). *Systems of logic based on ordinals: a dissertation*. Ph.D. Dissertation. Cambridge.

von Ahn, L. (2006). Games with a purpose. *Computer* 39, 6 (2006), 92–94.

von Ahn, L and Dabbish, L. (2004). Labeling images with a computer game. In *Proc. 22nd SIGCHI Conf. on Human factors in computing systems*. ACM, 319–326.

von Ahn, L, Liu, R, and Blum, M. (2006). Peekaboom: a game for locating objects in images. In *Proc. 24th SIGCHI Conf. on Human factors in computing systems*. ACM, 55–64.

von Ahn, L, Maurer, B, McMillen, C, Abraham, D, and Blum, M. (2008). reCAPTCHA: human-based character recognition via web security measures. *Science* 321, 5895 (2008), 1465–1468.

Wallace, B, Small, K, Brodley, C, and Trikalinos, T. (2011). Who Should Label What? Instance Allocation in Multiple Expert Active Learning.. In *SDM*. 176–187.