CrowdForge: Crowdsourcing Complex Work

Aniket Kittur, Boris Smus, Robert E. Kraut

February 1, 2011
CMU-HCII-11-100

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract: Micro-task markets such as Amazon's Mechanical Turk represent a new paradigm for accomplishing work, in which employers can tap into a large population of workers around the globe to accomplish tasks in a fraction of the time and money of more traditional methods. However, such markets typically support only simple, independent tasks, such as labeling an image or judging the relevance of a search result. Here we present a general purpose framework for accomplishing complex tasks using micro-task markets. Our approach is inspired by the MapReduce framework for distributed processing and provides a scaffolding for complex human computation tasks. We describe our general framework, a web-based prototype, and case studies on article writing and decision making that demonstrate the benefits of the approach.

This work has been funded in part by the National Science Foundation under grants OCI-0943148 and IIS-0968484. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the sponsors.

Keywords: crowdsourcing, Mechanical Turk, human computation, distributed processing, MapReduce, coordination.

INTRODUCTION

Crowdsourcing has become a powerful mechanism for accomplishing work online. Hundreds of thousands of volunteers have completed tasks including classifying craters on planetary surfaces (clickworkers.arc.nasa.gov), deciphering scanned text (recaptcha.net), and discovering new galaxies (galaxyzoo.org). Crowdsourcing has succeeded as a commercial strategy for accomplishing work as well, with companies accomplishing work ranging from crowdsourcing t-shirt designs (Threadless) to research and development (Innocentive).

One of the most interesting developments is the creation of general-purpose markets for crowdsourcing diverse tasks. For example, in Amazon's Mechanical Turk (MTurk), tasks range from labeling images with keywords to judging the relevance of search results to transcribing podcasts. Such "micro-task" markets typically involve short tasks (ranging from a few seconds to a few minutes) which users self-select and complete for monetary gain (typically from 1-10 cents per task). These markets represent the potential for accomplishing work in a fraction of the time and money required by more traditional methods [5][12][19][21].

However, the types of tasks accomplished through MTurk have typically been limited to those that are low complexity, independent, and require little time and cognitive effort to complete. Here we describe a framework for extending micro-task markets to support complex, interdependent tasks. Our framework is based on insights from the organizational behavior literature for understanding the kinds of coordination dependencies involved in complex tasks, and from the distributed computing literature for techniques to support those dependencies in distributed human computation. We make three primary contributions:

- We identify the coordination requirements necessary to crowdsource complex tasks, and describe a framework to support a variety of task types. The framework systematically breaks complex problems down into simpler tasks by creating subtasks that define and create other subtasks and distributes these tasks to workers. Output from subtasks can be evaluated and consolidated via additional outsourced tasks.
- We instantiate the framework in a prototype system for architecting the problem and managing the solution flow. We demonstrate the utility of the system through case studies using the prototype to solve complex problems including writing an article and researching a purchasing decision. We show quantitative evidence that the approach works better than asking a single individual to solve the problem, even controlling for the total wages offered.
- We demonstrate how the system can extend to more complex flows that include verification phases, and provide evidence that such phases can lead to better outcomes. We also describe how the system can be extended to support arbitrarily complex and recursive flows.

MECHANICAL TURK

According to Amazon, Mechanical Turk is "marketplace for work that requires human intelligence." Using it, employers post tasks, which Amazon terms "Human Intelligence Tasks" or "HITs." These are self-contained tasks that worker can select, work on, submit an answer, and collect a reward for completing.

There has been an increasing amount of research on Mechanical Turk recently. A large number of studies have examined the usefulness of MTurk as a platform for collecting and evaluating data for applications including machine translation [4] image databases [7], and natural language processing [19]. There have also been studies examining the use of MTurk as a platform for experiments in perception [5][10], judgment and decision making [11], and user interface evaluation [12]. Sorokin & Forsyth use Mechanical Turk to tag images in computer vision research and suggest voting mechanisms to improve quality [21]. Little et al. [15] introduced TurkIt, a toolkit for iterative tasks in MTurk, which we will discuss in more detail later in the paper.

MTurk encompasses a large and heterogeneous pool of tasks and workers; Amazon claims over 100,000 workers in 100 different countries, and as of the time of writing there were approximately 80,000 HITs available. The typical task on MTurk is self-contained, simple, repetitive, and short, requiring little specialized skill and often paying less than minimum wage. For example, during February 2010, we scraped descriptions of 13,449 HIT groups posted to Mechanical Turk. The modal HIT was allotted 60 minutes for completion and paid $0.03 US. We should note that the allotted time is often much greater than the actual completion time; assuming that workers received the US minimum wage of $7.25/hour or 12 cents per minute, a wage of $.03 per task implies that many tasks take a fraction of a minute to complete. Examples of tasks include identifying objects in a photo or video, de-duplicating data, transcribing audio recordings, or researching data details.

There are a number of reasons why these kinds of tasks are so common. Most workers (who call themselves "turkers") spend less than 8 hours per week on MTurk, and use it as a way to convert free time into a supplementary source of income, for pocket money, as a hobby, or to kill time, but not as their primary job [9]. As a result most turkers will not engage in long or complex tasks. Another potential explanation is the reward structure of Mechanical Turk, in which employers can reject work for any reason without giving an explanation. For example, to deal with spam, many employers summarily reject submitted work if it doesn't agree with others' submissions. Workers are unlikely to engage in long or complex tasks if there is a high probability that they will not get paid even if they completed the work in good faith. These rejections can have a serious impact on future earning too, since employers often ban workers whose prior work has been rejected over some threshold. This results in MTurk being a crowdsourcing market in which the average capacity for long or complex work is quite low.

COORDINATION IN COMPLEX TASKS

In contrast to the typical tasks posted on Mechanical Turk, much of the work required in many real-world work organizations and even many temporary employment assignments is often more complex, interdependent, and requires significant time and cognitive effort [16]. These tasks require substantially more coordination among co-workers than do the simple tasks typically seen on micro-task markets. The impact of micro-task markets would be substantially greater if they could also be applied to these more complex and interdependent task.

Consider for example the task of writing a short article about a locale for a public relationship campaign, a newspaper, a travel guide or an encyclopedia. This is a complex and highly interrelated task that involves many subtasks, such as deciding on the scope and structure of the article, finding and collecting relevant information, writing the narrative, taking pictures, laying out the document and editing the final copy. Furthermore, if more than one person is involved in the task, each person needs to coordinate in order to avoid redundant work and to make the final product coherent. Many kinds of tasks ranging from researching where to go on vacation to planning a new consumer product to writing software share the properties of being complex and highly interdependent, requiring substantial effort from individual contributors.

These difficulties with coordinating complex tasks are not unique to crowdsourcing. Organizational scholars have long investigated coordination across individuals and parts of a firm. Williamson won a Nobel Prize in economics for his theory of transaction costs [24], identifying the conditions under which it is efficient for a company to buy goods and services on a market instead of hiring employees to produce the work in house. According to this view, markets are efficient only when the costs of identifying an appropriate supplier, defining the exchange (e.g., the nature of the goods to be produced and its price), policing and enforcing the contract and settling the transaction are less than can be accomplished if the transaction were conducted in-house. Using employers may reduce transactions costs because at employees at least partially identify with the interests of the company and agree to be supervised and directed by managers. In an early paper, Malone and his colleagues argued that electronic markets may reduce the costs of search, bargaining, enforcement and settlement and dramatically expand the types of goods and services that can be offered on electronic markets [17]. The goal of our research to continue to reduce these transaction costs, by disaggregating and re-aggregating complex, multi-person tasks so that task definition, production and supervision can all be done in micro-task markets.

Coordination Dependencies	**Organizational science**	**Distributed computing**
Shared resources	Resource allocation	Mapping tasks to processors
Producer/ Consumer	Prerequisite constraints	Managing serial computations
	Transfer	Distributing data (transfer)
	Usability	Distributing data (format)
Task/subtask	Top-down task goals	Parallelizing computations
	Dynamic subtask generation	

Table 1. Types of coordination dependencies relevant to crowdsourcing complex tasks (from [16]) and concepts they map to from organizational science and distributed computing.

Task partitioning with division of labor is one of the most commonly used methods for coordinating tasks. Adam Smith in his classic *The Nature and Causes of the Wealth of Nations* [22] described the efficacy benefits associated with it. Via division of labor, a greater pool of agents can work in parallel, the agents can be specialized for the tasks they perform, and they can complete an assignment without losing time switching from task to task [2].

This approach works well when the tasks involve what Van de Ven et al. [23] call “pooled interdependence”, in which the group product is a simple aggregation of the individually produced products. Such tasks are typical of those currently supported by Mechanical Turk, such as judging the relevance of a search result or finding company information on a web. However, aggregating the results of individually produced tasks does not work as well when the work is complex and the component tasks are interdependent. Splitting a interdependent project into subtasks incurs costs: those subtasks need to be assigned to workers suited to the task; workers may need to be trained; timing of subtasks may need to be coordinated; the results of one subtask may be the input to other subtasks; the approach taken in one subtask may constrain the work taken in another; standards and quality may need to be enforced across subtasks; subtasks may need to be integrated and made consistent; and so forth. These are the sorts of conditions that Williamson suggested make some types of work difficult to accomplish efficiently in a market.

Many of these challenges are exacerbated in crowdsourcing markets. Because individual participants in micro-task markets do not want to and may not be capable of engaging in long or complex tasks, work must be broken down into many small pieces that can be quickly accomplished, leading to high coordination costs for interdependent workers. Tasks must be simple enough for workers to easily learn and complete. More generally, the nature of the workers and their relationship with the company posting the work produces additional challenges. With low commitment by companies and high turnover among the workers, there is minimal opportunity for training. Current micro-task markets have little built-in support for managing the flow of tasks, either in terms of time or input-output, and workers generally complete tasks independently with no knowledge of what others have done, making it difficult to enforce standards and consistency.

To identify the types of coordination we will need to support for crowdsourcing complex tasks we turn to previous work on coordination in organizational behavior [16][17][18][23]. In particular, Malone & Crowston [16] suggest a useful taxonomy of interdependencies that require coordination that we draw on in our current work. These dependencies are summarized in Table 1 and described in more detail below:

Managing shared resources. Whenever a limited resource needs to be shared, coordinating resource allocation becomes important. Allocating a fixed pool of workers to multiple tasks that must be completed under a deadline is a classic example of managing shared resources. In producing a newsletter under a deadline, for example, the fixed staff of writers, editors and proof readers need to be allocated so that all the planned stories are written, and all of them don’t demand work from an overtaxed editing and proofing staff at the last minute. Moreover, unless all subtasks are known beforehand, this allocation could change depending on how tasks are split up. Therefore a dynamic task allocation process is needed to manage worker resources throughout the task flow.

Managing producer/consumer relationships. In many situations one activity produces something required as input for another activity. For example, the structure of an article needs to be decided on before the sections can be written. This leads to a number of dependencies: 1) prerequisite constraints: certain activities need to take place before others; 2) transfer: the output of one activity needs to be shared with another; 3) usability: the output of one activity needs to be in such a form that it can be used by another. For successful coordination in complex tasks each of these dependencies must be supported.

Managing task/subtask dependencies. There are many different ways a task can be divided into subtasks [18]. In a top-down approach, a single person (e.g., the task creator) could specify all of the subtasks (and sub-subtasks, etc.) to decompose a task into. However, in many cases decomposition of the task may itself be a part of the process. For example, the task creator may not be able to a priori specify all the sections for an article or the classes for a software program in advance. Supporting both top-down specification of the task goals and bottom-up identification of subtasks is crucial for crowdsourcing complex tasks in which subtasks may be generated and completed dynamically.

APPROACH

Our goal is to support the coordination dependencies involved in complex work through micro-task markets. As previously mentioned, most tasks on these markets are simple and self-contained with no challenging coordination dependencies. The audio transcription tasks posted by Castingwords.com are a rare exception to the typical MTurk task. Castingwords breaks up an audio stream into overlapping segments, and workers are employed to generate transcriptions from each audio segment. These transcriptions are then verified by other workers, whose work is later automatically put together into a complete transcription. Unlike the standard micro-market task, the disaggregation of an audio file into smaller transcription tasks and the use of a second wave of workers to verify the work done by the transcriptions involves the producer/consumer dependency and others of the dependencies identified in the previous section. It also provides a simple model for many of the elements of our approach. For example, the transcription task can be broken up into the following elements:

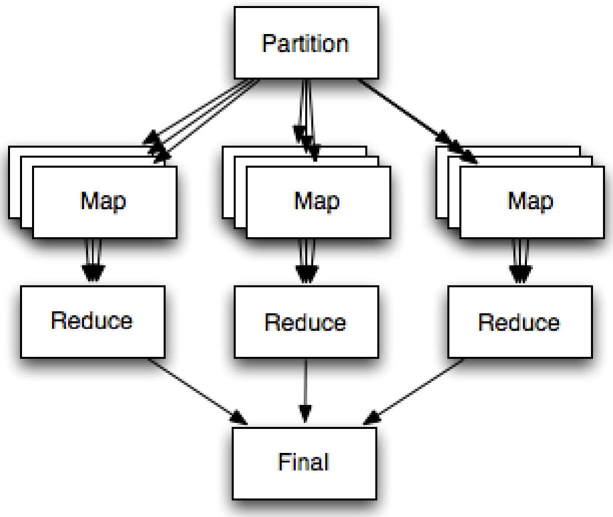


Figure 1: Overview of framework for splitting up and recombining complex human computation tasks.

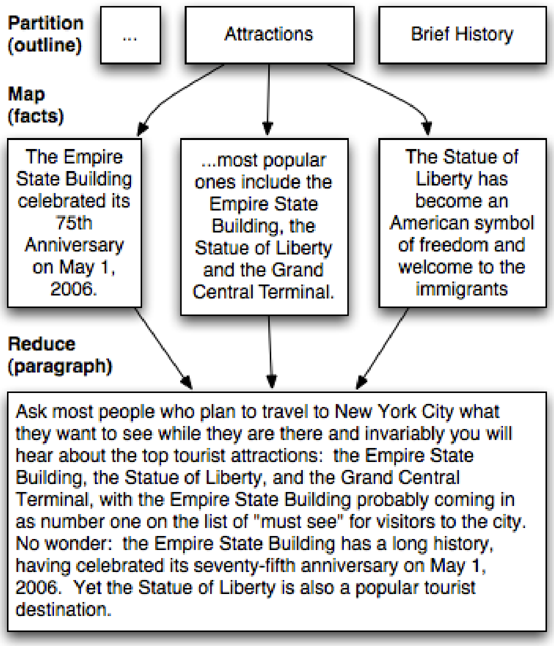
- A pre-specified partition that breaks up the audio into smaller subtasks
- A flow that controls the sequencing of the tasks and transfer of information between them
- A quality control phase that involves verification of one task by another worker
- Automatic aggregation of the results

The TurkIt toolkit for MTurk [15] extends some of these elements by enabling task designers to specify iterative flows. Little and colleagues use as an example a text identification in which the results of multiple workers' outputs are voted on and the best sent to new workers, whose work is then voted on, and so forth. Viewed according to the elements above, this is a producer/consumer dependency, with a flow between a production task (text identification) and a quality control task (voting) that also serves to aggregate the results.

Our goal is to generalize these elements into a framework that supports the crowdsourcing of highly complex work. Specifically, our framework aims to support:

- Multi-level partitions in which a task can be broken up by more than one partition
- Dynamic partitioning so that workers themselves can decide how to partition a task, with their results generating new subtasks during the flow (rather than the task designer needing fully specify partitioning beforehand)
- Complex flows involving many tasks and many workers
- A variety of quality control methods including voting, verification, or merging items
- Intelligent aggregation of results both automatically and by workers.
- A simple method for specifying and managing tasks and flows between tasks

To accomplish these goals our approach draws on concepts from the distributed computing literature [1][20]. Malone and Crowston [16] note the general parallels



Partition (outline)
...
Attractions
Brief History
Map (facts)
The Empire State Building celebrated its 75th Anniversary on May 1, 2006.
...most popular ones include the Empire State Building, the Statue of Liberty and the Grand Central Terminal.
The Statue of Liberty has become an American symbol of freedom and welcome to the immigrants
Reduce (paragraph)
Ask most people who plan to travel to New York City what they want to see while they are there and invariably you will hear about the top tourist attractions: the Empire State Building, the Statue of Liberty, and the Grand Central Terminal, with the Empire State Building probably coming in as number one on the list of "must see" for visitors to the city. No wonder: the Empire State Building has a long history, having celebrated its seventy-fifth anniversary on May 1, 2006. Yet the Statue of Liberty is also a popular tourist destination.

Figure 2: Partial results of a collaborative writing task.

between coordination in human systems and computer systems. Crowdsourced labor markets can be viewed as large distributed systems in which each person, such as a worker on Mechanical Turk, is analogous to a computer processor that can solve a task requiring human intelligence. In this way a crowdsourcing market could be seen as a loosely coupled distributed computing system [1].

Key challenges in distributed computing include partitioning computations into tasks that can be done in parallel, mapping tasks to processors, distributing data to and between processors, and fault tolerance [1][20]. Many of these challenges map to the coordination dependencies identified earlier, as shown in Table 1[1]. This suggests that some solutions to managing distributed computing with multiple processors may be profitably applied to crowdsourcing as well.

A simple, well-known, and widely used distributed computing model is Google's MapReduce framework [6]. MapReduce was inspired by functional programming languages in which a large array of data is processed in parallel through a two step process: first, key/value pairs are each processed to generate a set of intermediate key/value pairs (the Map phase). Next, values with identical intermediate keys are merged (the Reduce phase). Although there are many other models of distributed computing [1][20], key advantages of MapReduce include: 1) its simplicity, allowing programmers without experience in parallel programming to easily make use of distributed system resources; 2) a large variety of real world problems have been found amenable to representation in Map and Reduce tasks.

Our approach builds on the general approach to distributed computing exemplified by MapReduce of breaking down a complex problem into a sequence of simpler subtasks using a small set of subtask types (e.g., Maps and Reduces). We define three types of subtasks:

- Partition tasks, in which a larger task is broken down into discrete subtasks
- Map tasks, in which a specified task is processed by one or more workers
- Reduce tasks, in which the results of multiple workers' tasks are merged into a single output

CrowdForge abstracts away many of the programming details of creating and managing subtasks by treating partition/map/reduce steps as the basic building blocks for distributed process flows, enabling complex tasks to be broken up systematically and dynamically into sequential and parallelizable subtasks.

In partition tasks, workers are asked to create a high level partitioning of the problem, such as creating an outline of an article with section headings or a list of criteria for buying a new car. In our system the partitioning is made an explicit part of the task itself, with subtasks dynamically created based on the results of the partition step. Importantly, this means that the task designer does not have to know beforehand all of the subtasks that will be generated: defining the division of labor and subtask design are shifted to the market itself.

In map tasks, a specified processing step is applied to each item in the partition. These tasks are ideally simple enough to be answerable by a single worker in a short amount of time. For example, a map task for article writing could ask a worker to collect one fact on a given topic in the article's outline. Multiple instances of a map tasks could be instantiated for each partition; e.g., multiple workers could be asked to collect one fact each on a topic in parallel.

Finally, reduce tasks take all the results from a given map task and consolidate them, typically into a single result. In the article writing example, a reduce step might take facts collected for a given topic by many workers and have a worker turn them into a paragraph.

Any of these steps can be iterative. For example, the topic for an article section defined in a first partition can itself be partitioned into subsections. Similarly, the paragraphs returned from one reduction step can in turn be reordered through a second reduction step.

CASE STUDIES

Before describing implementation details of the system we first provide examples and evidence of how the system works in practice through two case studies: article writing and researching a purchase decision.

Article writing

The first complex task we explored was writing an encyclopedia article. Writing an article is a challenging and interdependent task that involves many different subtasks: planning the scope of the article, how it should be

[1] Fault tolerance is not included in the table as it does not map well to the identified coordination dependencies. However, we will address this issue in more detail in the section on quality control below.

structured, finding and filtering information to include, writing up that information, finding and fixing grammar and spelling, and making the article coherent. While there are examples of collaborative writing on the Internet, notably Wikipedia, previous work has shown that the success of harnessing a large group of contributors is often dependent on a small core of leaders that do a large proportion of the work and organize the contributions of others [13][14]. This poses a challenge in micro-task markets where individuals may not be willing to spend the large amount of effort needed to be a leader and may not be able to communicate with others in order to coordinate or influence their behavior. Furthermore, because of the many related subtasks, article writing encompasses many of the coordination dependencies in Table 1. Finally, many of the subtasks involved, such as assembling the relevant information or doing the actual writing, can be time consuming and complex. These characteristics make article writing a challenging but representative test case for our approach.

To solve this problem we created a simple flow consisting of a partition, map, and reduce step. The partition step asked workers to create an article outline, represented as an array of section headings such as "History" and "Geography". In an environment where workers would complete high effort tasks, the next step might be to have someone write a paragraph for each section. However, the difficulty and time involved in finding the information for and writing a complete paragraph for a heading is a mismatch to the low work capacity of micro-task markets. Thus we broke the task up further, separating the information collection and writing subtasks. Specifically, each section heading from the partition was used to generate map tasks in which multiple workers were asked to submit a single fact about the section (turkers were also asked to submit a URL reference to the source of the fact to encourage high quality fact collection).

Next, the reduction step asked other workers to create a paragraph for each section based on the facts collected in the map step. By separating the collection of information and writing into two stages we could significantly decrease the cost of each stage, making the task more suitable for micro-task workers. In addition, we benefit from other effects such being able to collect more and more diverse information when more workers were involved. The partition-map-reduce process is illustrated in Figure 2. Finally, since the sections of encyclopedic articles are relatively independent, the resulting paragraphs were themselves reduced into an article by simply concatenating them[2].

We used this approach to create five articles about New York City. Articles cost an average of $3.26 to produce, and required an average of 36 subtasks or HITs, each performed by an individual worker. Partition-workers identified 5.3 topics per article in the partition step. The average number article ended with 658 words. A fragment of a typical article is shown in Figure 2; this article consisted of 955 words and 7 sections: brief history, getting there, basic layout, neighborhoods, getting around, attractions and ethnic diversity. It was completed via 36 different HITs for a total cost of $3.15.

To verify the quality of these collaboratively written articles, we compared them to articles written individually by workers and to the entry from the Simple English Wikipedia on New York City [25]. To produce a comparison group of individually written articles, we created eight HITs which each requested one worker to write the full article. To control for motivations associated with reward, we paid these individuals $3.05, approximately the same amount as the average group payment. The resulting articles consisted of an average of 393 words, approximately 60% the length of the collaborative written articles.

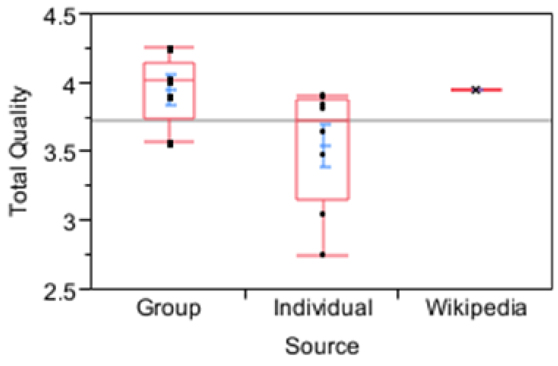


Figure 3: Rated quality of articles about New York City produced by Mechanical Turk workers acting individually or as a group using our framework compared to the quality of the same article on the Simple English Wikipedia

We then evaluated the quality of all articles by asking a new set of workers to each rate a single article based on four dimensions: use of facts, spelling and grammar, article structure, and personal preference. Fifteen workers rated each article on five-point Likert scales. We averaged the ratings of the 15 raters across the four dimensions to get an overall quality score for each article.

On average the articles produced by the group were of higher quality than those produced individually (see Figure 3: mean quality for group-written articles = 4.01

[2] For other kinds of articles there could be another crowdsourced reduce phase that integrates the paragraphs.

versus 3.75 for individually-written ones, *t*(11)=2.17, *p*=.05).

The average quality for the group-written articles was roughly the same as the Simple English Wikipedia article (Wikipedia quality=3.95). Not only was the average quality of the group articles higher than the individually written ones, but as Figure 3 also shows, the variability was lower as well (*t*(11)=-2.43, p=.03*)*, with a lower proportion of poor articles.

Overall, we found that using CrowdForge to crowdsource the complex and interdependent task of article writing worked surprisingly well. Despite the coordination requirements involved in managing and integrating the work of dozens of workers, each contributing only a small amount of work, the group-produced articles were rated higher and had lower variability than individual-produced articles -- even though individuals were paid the same amount as the whole group and did not have to deal with coordination challenges -- and similar in quality to corresponding Simple Wikipedia articles.

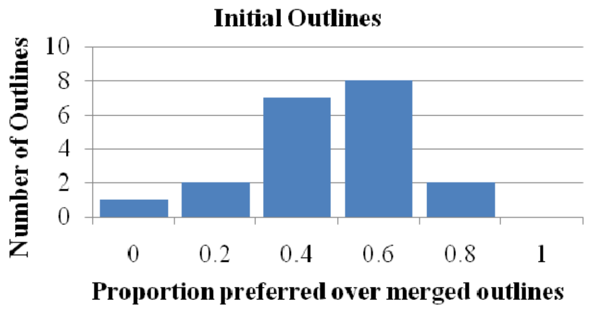
Quality Control

In the article writing study each partition task was completed by a single worker. This creates the possibility that a single bad partition (i.e., outline) could have a large negative effect on the whole task. We found this did occur in one of the group articles, with a bad outline's effects cascading down the task chain. It is remarkable that despite this brittleness, we still found a robust advantage of the group condition over the individual condition, speaking to the strength of the approach. However, in many cases we would like to minimize the likelihood of any task failing due to a single low quality worker by combining multiple workers' results.

Our approach to dealing with this challenge is to utilize additional Map or Reduce tasks to supporting fault tolerance and quality control. For example, to represent Castingwords transcription flow, described earlier, in the CrowdForge framework, workers verifying the results of other workers' outputs can be represented as a Map task that applies a verification function to each value. Other kinds of quality control processes can also be applied; for example, voting on the best choice can be represented as a Reduce task in which a single output is chosen from multiple workers' outputs based on the vote. Other kinds of human intelligence tasks could also be used, such as a Reduce task in which workers combine the best aspects of other workers' outputs rather than choosing a single best output. An advantage of this approach is that quality control steps are treated the same as other kinds of subtasks, minimizing added complexity. This also means that many different kinds of quality control can be used depending on the task and the kind of fault tolerance desired, as opposed to forcing task designers to use a single mechanism such as voting or verification.

To test the utility of different quality control methods in the article writing case we ran an additional experiment on MTurk. We were especially interested in whether more complex quality control methods that require human intelligence -- such as combining the best aspects of multiple workers' outputs -- would work better than methods such as simple voting. Merging results (in this case, article outlines) could have a number of advantages over voting. The likelihood of a poor outline could be reduced, since at the very least – if workers did no merging at all – the best of the outline options should be chosen. It is also possible that the merged outlines could be better than the initial outlines, if the best aspects of each of multiple outlines were combined, or if seeing multiple outlines at once facilitates comparison between them and thus leads to better outlines.

To test these hypotheses we ran an experiment on quality control of article outlines. In the first phase we asked 20 turkers to each independently generate an outline for an article on the recent Gulf of Mexico oil spill using the same procedure as in the article writing case study above. We then took these 20 outlines and randomly assigned them into 20 different sets of three outlines (outlines could be in more than one set). Each set was given to a different turker, who were asked to



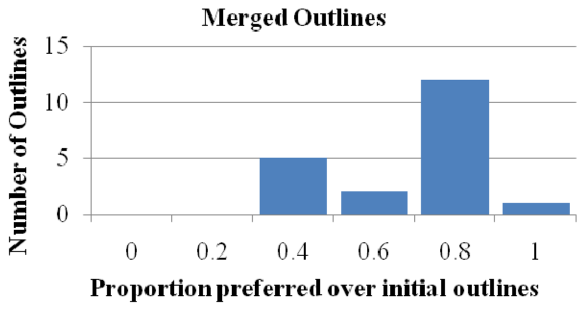


Figure 4. Histograms of participants preference choices for initial and merged outlines.

create a new outline for the article using elements from the 3 outlines in his or her set (i.e., a Reduce task). Turkers were instructed to use any elements from any of the outlines they had available, but were not allowed to add new elements. This resulted in 20 merged outlines.

For evaluation we crossed the 20 initial and 20 merged outlines, resulting in 400 paired comparisons between an original outline and a merged one. We then asked turkers to choose which outline in each comparison would result in a better article. To ensure that turkers evaluated both outlines, they were also required to identity which elements in the two outlines were the same, with the caveat that elements did not need to be worded exactly the same in the two outlines as long as they were conceptually similar. This question follows the best practices outlined in [12] for subjective tasks, by using a verifiable question that forces the worker to do much of the processing they would need to in order to make a good evaluation: by comparing the two outlines, they could then evaluate factors such as which elements were missing from one outline, and the level of cohesion of the different elements. An outline's quality score is the number of comparisons that it won.

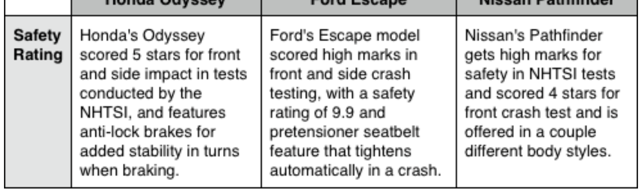
Results showed that the merged outlines were rated as better outlines more often than the initial outlines: 61% of merged outlines were chosen compared to 39% of initial outlines. A binomial test revealed the results as statistically significant ($p < .001$). Histograms of choice preferences for individual outlines are shown in Figure 4. The results indicate that merged outlines had fewer poor outlines: while 7 of the initial outlines were preferred 35% or less of the time, no merged outlines had preference values lower than 35%. Furthermore, the best merged outlines were considered better than the best initial outlines: the best initial outline was preferred 74% of the time, while 3 merged outlines were preferred more than that with the highest preferred 90% of the time. Together these results suggest that complex quality control tasks such as combining the results of multiple workers' outputs can be more effective than simple voting. CrowdForge makes such quality control tasks simple to implement as Reduce tasks.

Researching a purchase

We also investigate a different task— researching purchase decisions—in order to test the generality of the framework. Modern purchasing decisions can be overwhelming because of large numbers of choices and information about each choice. Though there is a lot of information available to help a consumer make intelligent purchasing choices, it is often distributed across unrelated forums and websites across the Internet. Purchase decisions could benefit from an intelligent, personalized processing of the options. We applied our framework to commission decision matrices, in this case to help consumers compare automobiles.

The previous example described a unidimensional partition process for accomplishing complex work. This example extends the framework by showing how one can partition the initial task on multiple dimensions (or, equivalently, repartition each element of the original partition). In the partition HIT for this problem, one worker was given a short description of a consumer in the market for a car (a hypothetical suburban family that drives their two children to and from school and enjoys occasional road trips) and asked to submit criteria for an automotive purchase that would be relevant to that persona. Another worker was given the same description and requested to submit a list of potential cars that might be interesting for the persona to buy. Combining the resulting lists yielded a matrix resembling a blank product comparison table, in which for example, a Honda Odyssey could be compared to a Ford Escape on dimensions such as reliability and safety. In the map step, workers were asked to submit facts for one cell in the table, for example evidence relevant to safety ratings for the Honda Odyssey they might find from an online review of the car. Finally, in the reduce step workers were given all the facts for a cell collected by workers in the map step, and were asked to write a single sentence consolidating them.

We used this approach to commission a personalized car purchase recommendation table for the hypothetical family described above. In the partition step we asked some workers to submit potential car make/models relevant to the family, and others to submit purchasing criteria they thought the family might employ. For simplicity, we accepted the three most commonly cited items from each dimension: the cars Honda Odyssey, Ford Escape and Nissan Pathfinder crossed with the evaluation criteria room, price and safety rating. Other workers then found relevant facts for each cell, and yet others consolidated those facts into a sentence according to the process described above. The entire task was completed in 54 different HITs for a total cost of \$3.70. An excerpt from the resulting product



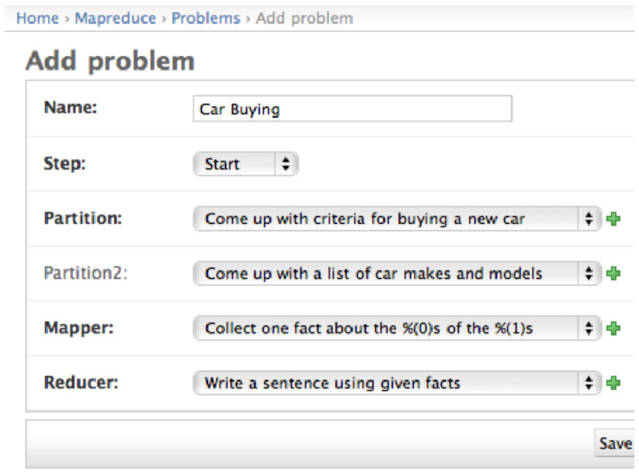
	Honda Odyssey	Ford Escape	Nissan Pathfinder
Safety Rating	Honda's Odyssey scored 5 stars for front and side impact in tests conducted by the NHTSI, and features anti-lock brakes for added stability in turns when braking.	Ford's Escape model scored high marks in front and side crash testing, with a safety rating of 9.9 and pretensioner seatbelt feature that tightens automatically in a crash.	Nissan's Pathfinder gets high marks for safety in NHTSI tests and scored 4 stars for front crash test and is offered in a couple different body styles.

Figure 5: An excerpt from the product comparison table

comparison table is shown in Figure 5. While the CrowdForge frame was successful in producing the comparison, table, we had no success getting even a single worker to generate a similar product comparison chart individually, even when offering more money than we paid the entire group.

PROTOTYPE

We implemented a software prototype to test our approach by allowing task designers to indirectly use MTurk to solve complex problems. It was this prototype that generated the HITs used in the experiments described above. The prototype allows task designers to break complex problems down into sub-problems, to specify the relationship between the sub-problems, and to generate a solution using MTurk. The system consists of a web user interface for the task designer, and a backend server which interfaces with Amazon's MTurk servers. The web user interface in Figure 6 allows users to define each step in the problem solving process and to specify the flow between each step. The server-side component creates MTurk HITs, consumes their results, and generates new HITs as needed. The prototype is written in Python using Django [8], a high-level web framework for rapid application development. Boto [3], a Python interface to Amazon Web Services, is used to communicate with MTurk.



Add problem

Name:	Car Buying
Step:	Start
Partition:	Come up with criteria for buying a new car
Partition2:	Come up with a list of car makes and models
Mapper:	Collect one fact about the %(0)s of the %(1)s
Reducer:	Write a sentence using given facts

Figure 6: Creating a problem with the web user interface.

The system abstracts the entire process as a *problem*, which references multiple *HIT Template*s (which may be either partitions, maps, or reduces), and a *flow* that defines the dependencies between the HIT Templates. The prototype allows multiple problems to exist in parallel, each one tracking its own currently active HIT Template. HIT Templates are parameterized templates used to create HITs on MTurk, specifying basic parameters like title, HTML body and compensation amount. Finally, flows manage the sequential coordination between HITs, as well as transferring data between HITs.

The system uses a notification-based flow control mechanism to manage which tasks and templates are posted. Every few minutes the system monitors active problems for four kinds of events, and fires notifications as needed. The *result retrieved* notification fires when the system detects a new result from MTurk. The *HIT expired* notification fires when a HIT that was posted by the prototype expires due to the HIT lifetime running out. The *HIT complete* notification fires when all instances of a HIT were completed by turkers. The *stage complete* notification fires when all HITs of the currently active HIT Template are completed or expired.

The simplest predefined flow (Figure 1) starts with a partition HIT Template, the result of which is fed into a map HIT Template, the results of which are fed into a reduce HIT Template. Transitions between these three HIT Templates occur when the s*tage complete* notification fires. In the article writing example, this flow takes the article outline generated by a worker completing a partition HIT, and creates map HITs to collect facts for each heading in the outline. Note that this process is dynamic: the number of headings does not need to be specified beforehand by the task designer. Once all map HITs for a heading are complete, the flow posts a reduction HIT to consolidate all facts collected in the map HITs into a paragraph. The prototype has several such predefined flows, but also allows developers to write custom flows and register them with the system.

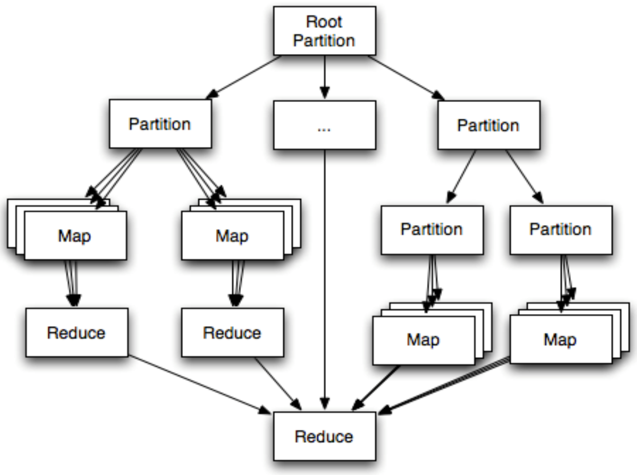


Figure 7. Nested subtasks forming a complex flow.

COMPLEX FLOWS

The example of article writing assumed a simple linear flow from partitioning to mapping to reduction.

This linearity may not be powerful enough to represent some tasks. For example, a textbook may have a complex outline more closely resembling a tree than a list. This hierarchical outline can be achieved by modifying the partition hit type: in addition to collecting headings, include a checkbox allowing workers to specify whether or not each heading should be further partitioned. As previously mentioned, the CrowdForge prototype allows developers to implement additional flows for more complex cases. If the turker specifies to partition a heading into sub-headings, the tree flow should create partition HITs. Otherwise, the tree flow should create map HITs as in the simple flow.

In the more general case, subtasks can be themselves be broken down into partition, map and reduce phases (Figure 7). The notification-based architecture of the CrowdForge prototype allows this kind of nesting to be implemented as a custom flow; future work will allow the creation of nested flows using the GUI tools only.

CONCLUSION

In this paper we presented a general-purpose framework for solving complex problems through micro-task markets. Based on concepts from coordination science and distributed computing, the CrowdForge framework provides a systematic and dynamic way to break down tasks into subtasks and manage the flow and dependencies between them. Key contributions of the framework include support for dynamic and multi-level partitions, complex flows, quality control, complex aggregation, a GUI for task designers, and a simple unifying framework for task and flow management. We demonstrate through two case studies how the framework can break down complex tasks such as writing an article or researching a purchase decision into flows of partition, map, and reduce subtasks. In the article writing case we showed that CrowdForge-produced articles were rated more highly and had lower variability than individual-produced articles, despite the coordination requirements of managing and integrating dozens of workers, and were rated of similar quality as Simple Wikipedia articles. In an extension to this example, we showed how to insert a quality control step in the flow and demonstrated the value of combining the best aspects of multiple workers' outputs rather than simple voting. In the purchase decision case we were unable to get even a single individual to complete the task given the high effort involved, but could accomplish the goal using CrowdForge with low monetary costs.

As general purpose markets continue to evolve, there is a growing need to be able to solve a wider range of tasks of increasing complexity and coordination requirements. Furthermore, as the nature of work itself becomes more distributed, such an approach has the potential to change the way that work gets done, enabling many more people to be involved in solving complex problems ranging from business intelligence to writing software. However, markets don't work well for complex tasks when the employer cannot define exactly what they want in advance or if the contract is difficult to pre-define. The CrowdForge framework reduces this need for predefinition by allowing for subtasks to be dynamically generated by the market itself.

There are a number of directions we are exploring for future work. Most immediately, one challenge is extending our GUI to support more complex, nested flows so that task designers with no programming experience can complete arbitrarily complex work that involves high coordination dependencies. Looking further ahead, we are interested in exploring the possibilities of the CrowdForge framework in other kinds of task markets. For example, if the framework was applied to a market in which the expertise of individual workers was known (e.g., in a corporation) there might be greater opportunities for managing resource allocation of workers to appropriate tasks. Feedback about the selection and quality of their past work could also be useful for improving resource allocation if the system had a shared memory of individual workers' history across tasks.

Our approach is also an example of how coordination solutions can cut across multiple fields. Coordination theorists have emphasized that some principles of coordination generalize across radically different entities being coordinated, as when principles from economics are used to schedule tasks for a computer operating system [16]. Here we tie coordination science to distributed computing, demonstrating how principles from the domain of coordinating computer systems can be used to coordinate human activity.

ACKNOWLEDGMENTS

This work has been funded in part by the National Science Foundation under Grants OCI-0943148 and IIS-0968484. We would also like to thank Susheel Khamkar for his valued contributions.

REFERENCES

[1] Bal, H.E., Steiner, J.G. and Tanenbaum, A.S. Programming languages for distributed computing systems. ACM Computing Surveys (CSUR) (1989) vol. 21 (3).

[2] Becker, G.S. and Murphy, K.M. The division of labor, coordination costs, and knowledge. The Quarterly Journal of Economics (1992) vol. 107 (4) pp. 1137-1160

[3] Boto, http://code.google.com/p/boto/

[4] Callison-Burch, C. Fast, cheap, and creative: evaluating translation quality using Amazon's Mechanical Turk. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1 (2009) pp. 286-295

[5] Cole, F., Sanik, K., DeCarlo, D., Finkelstein, A., Funkhouser, T., Rusinkiewicz, S., and Singh, M. How well do line drawings depict shape? In ACM SIGGRAPH (2009), 1–9.

[6] Dean, J. and Ghemawat, S. Map Reduce: Simplified data processing on large clusters. Communications of the ACM-Association for Computing Machinery-CACM, 51, 1 (2008), 107-114.

[7] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L. "ImageNet: A large-scale hierarchical image database," cvpr, pp.248-255, 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009

[8] Django, http://www.djangoproject.com

[9] Ipeirotis, P. (2010, Mar 3). The New Demographics of Mechanical Turk. http://behind-the-enemy-lines.blogspot.com/2010/03/new-demographics-of-mechanical-turk.html. Retrieved 9-21-20

[10] Heer, J. and Bostock, M. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. Proceedings of the 28th international conference on Human factors in computing systems (2010) pp. 203-212

[11] Horton, J. J., Rand, D.G. and Zeckhauser, R.J., The online laboratory: Conducting experiments in a real labor market (2010). NBER Working Paper Series, Vol. w15961.

[12] Kittur, A., Chi, E., Suh, B. Crowdsourcing User Studies With Mechanical Turk. In *Proceedings of CHI* (2008).

[13] Kittur, A., Lee, B., Kraut, R. E. Coordination in Collective Intelligence: The Role of Team Structure and Task Interdependence. In *Proceedings of CHI* (2009).

[14] Kittur, A. and Kraut, R. E. 2008. Harnessing the wisdom of crowds in wikipedia: quality through coordination. CSCW '08. ACM, New York, NY, 37-46

[15] Little, G., Chilton, L., Goldman, M. and Miller, R. TurKit: Tools for iterative tasks on mechanical turk. SIGKDD Workshop on Human Computation (2009).

[16] Malone, T. and Crowston, K. The interdisciplinary study of coordination. ACM Computing Surveys, 26, (1994), 87-119.

[17] Malone, T., Yates, J. and Benjamin, R. Electronic markets and electronic hierarchies. Communications of the ACM, 30 (6). 484-497.

[18] Mintzberg, H. 1979. The Structuring of Organizations.Prentice-Hall, Englewood Cliffs, N.J.

[19] Snow, R., O'Connor, B., Jurafsky, D., and Ng, A.Y. Cheap and fast---but is it good?: evaluating non-expert annotations for natural language tasks. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics (EMNLP 2008), 254-263.

[20] Skillicorn, D.B., Talia, D. Models and languages for parallel computation, ACM Computing Surveys 30(2) (1998) 123–169.

[21] Sorokin, A., Forsyth, D. Utility data annotation with Amazon Mechanical Turk, First IEEE Workshop on Internet Vision at CVPR 08.

[22] Smith, A. The Nature and Causes of the Wealth of Nations. Liberty Classics, Indianapolis:, 1776.

[23] Van de Ven, A., Delbecq, A. and Koenig, R. Determinants of coordination modes within organizations. American Sociological Review, 41 (1976), 322-338.

[24] Williamson, O. Transaction-Cost Economics: The Governance of Contractual Relations. The Journal of Law and Economics, 22 (2). 233.

[25] Wikipedia Simple entry on New York City, http://simple.wikipedia.org/wiki/New_York_City