Just in Time: Controlling Temporal Performance in Crowdsourcing Competitions

Markus Rokicki
L3S Research Center,
Hannover, Germany
rokicki@L3S.de

Sergej Zerr
Electronics and Computer Science, University of Southampton, Southampton, UK
s.zerr@soton.ac.uk

Stefan Siersdorfer
siersdorfer@outlook.de

ABSTRACT

Many modern data analytics applications in areas such as crisis management, stock trading, and healthcare, rely on components capable of nearly real-time processing of streaming data produced at varying rates. In addition to automatic processing methods, many tasks involved in those applications require further human assessment and analysis. However, current crowdsourcing platforms and systems do not support stream processing with variable loads. In this paper, we investigate how incentive mechanisms in competition based crowdsourcing can be employed in such scenarios. More specifically, we explore techniques for stimulating workers to dynamically adapt to both anticipated and sudden changes in data volume and processing demand, and we analyze effects such as data processing throughput, peak-to-average ratios, and saturation effects. To this end, we study a wide range of incentive schemes and utility functions inspired by real world applications. Our large-scale experimental evaluation with more than 900 participants and more than 6200 hours of work spent by crowd workers demonstrates that our competition based mechanisms are capable of adjusting the throughput of online workers and lead to substantial on-demand performance boosts.

General Terms

Algorithms, Experimentation, Human Factors

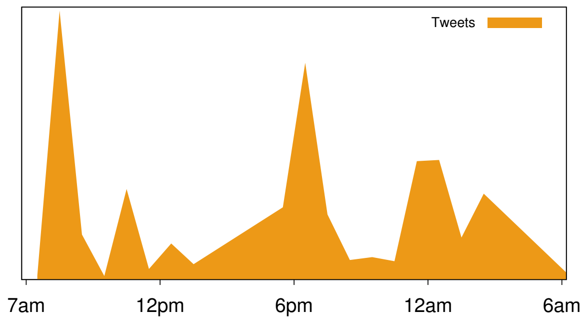
Keywords

crowdsourcing; stream processing; competitions.

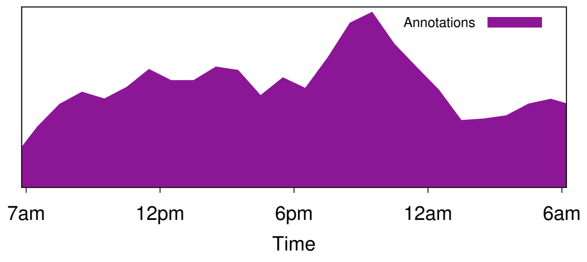
1. INTRODUCTION

Advances in efficient algorithms and computer hardware have enabled the processing of huge amounts of data collected from a variety of sources. This includes data continuously produced in streams such as output of digital sensors in production management, user generated content in online social networks, and stock trading data. These scenarios often require (near) real time processing of incoming data portions. Furthermore, in many cases, the rate of incoming data can largely vary over time, resulting in data peaks and temporarily large increases in processing demands. This holds, for instance, for scenarios such as crisis management, business analytics, or healthcare, that require immediate analysis and action during exceptional situations [29, 3].

In addition to automatic processing methods, many tasks involved in data analytics applications require further human assessment and analysis. Crowdsourcing has become more and more popular for efficiently acquiring human annotations and evaluations. However, platforms such as Amazon's Mechanical Turk and CrowdFlower cannot ensure that



(a) Tweet volumes



(b) Annotation output

Figure 1: April 2015 Nepal Earthquake tweet volume during first 24 hours vs. typical annotation throughput of a crowdsourcing system [31]. In this work we explore methods for adaptation to temporal changes in data volume and requirement characteristics in order to overcome such demand-supply gaps.

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
http://dx.doi.org/10.1145/2872427.2883075.

the workers' performance over time is in sync with demand based on the properties of incoming data. In many cases the average crowd performance at a given time is largely influenced by aspects such as the time zones of the workers processing the tasks, and is often subject to periodic fluctuations due to day-night cycles [30, 31].

As an example for an event monitoring activity, Figure 1 shows the hourly amount of the incoming Twitter messages (which might roughly correspond to the demand in a data analytics application) in the context of the Nepal earthquake in April 2015 along with the typical daily activity (offer) of the crowd as observed in previous crowdsourcing experiments [31]. The example illustrates two major issues: On one hand, a data analytics application might not receive enough support to process data during peak periods of the data stream. On the other hand, too many (possibly unnecessary and costly) annotations might be produced during periods, where the offer exceeds the demand.

For *automatic* data stream processing, many efficient algorithms and models were proposed in the literature including but not limited to representative stream sampling [20], streaming item diversification [24], or stream clustering [37]. The performance of such algorithms is typically well predictable and is influenced on the one hand by the available hardware and on the other hand by stream properties such as volume, velocity and variety. Stream processing management systems such as QualiMaster[1] can adapt to dynamic changes of incoming stream properties and choose between alternative algorithms with similar functionality and different running time properties. The algorithms are typically evaluated in terms of their quality, throughput, and latency. Whilst it has been shown that also *crowd workers* can be motivated to process higher amounts of data through both monetary and non-monetary incentives [23, 30, 31, 11], adaptation of crowd performance to dynamic changes of the demand and the resulting ability of crowdsourcing platforms to handle streaming data has largely remained unexplored.

In this work we present a large scale study providing insights into streaming properties of crowdsourcing platforms. To this end, we introduce a temporal utility framework for crowdsourcing as well as methods for controlling the temporal behavior in crowdsourcing with respect to properties of the streaming data. In particular we examine how fast the crowd can react to demand changes and how workers respond to dynamic incentive mechanisms within crowdsourcing competitions. We evaluate the performance for different demand scenarios and incentive-related settings using a state-of-the-art competition based crowdsourcing framework for maximizing the output whilst bounding the costs.

Outline. The remainder of this paper is organized as follows: In Section 2 we discuss related work on crowdsourcing incentives, pricing schemes, and time dependent demands. In Section 3 we introduce a temporal utility-based framework for crowd output and describe our strategies for time aware control of crowd behavior. The evaluation of our strategies is presented in Section 4 where we first describe the experimental setup along with the core results, and then delve deeper into details about worker behavior and performance. Finally, in Section 5 we conclude and describe directions of our future work.

[1]`http://qualimaster.eu`

2. RELATED WORK

There is a body of work considering temporal evolution of the incoming data in different scenarios. In [16] a general framework for crowdsourced stream processing systems (CSPs) is proposed where crowd workers validate the output of machine based processing. The same authors apply their methods for detecting informative tweets in the context of disaster management [15, 14]. They also elaborate on changes in the class distributions within streams and the requirement of frequently updating classifier using labels gathered over time. However, the evaluation is just focused on classifier performance and does not consider the time needed for the crowd to adopt to the changes. For shortening the response times, standby worker pools were introduced in [4, 2] and adjusting query execution plans to reduce the number of crowdsourced tasks was suggested in [9]. For crowd supported item search, Das Sarma et al. [7] explore tradeoffs between cost and latency, considering the number of crowdsourcing iterations over time. Based on observations at Amazon's Mechanical Turk, Faridani et al. [10] propose a theoretical framework for the relationship of HIT price and completion time to speed up processing. They identify the size of task batches along with the monetary reward per batch as important factors for attracting workers. Building on that work Gao and Parameswaran [12], investigate static and dynamic variants of pricing schemes to achieve desired completion time with high probability. Difallah et al. [8] employ dynamic bonus rewards at fixed intervals to decrease completion time by motivating top contributors to stick to a given task.

None of these works is dealing with real-time requirement changes on aggregate crowd performance. In contrast, in our work, we especially focus on scenarios where the crowd has to adjust annotation throughput, reacting to sudden changes on demand.

Time-based pricing is a strategy widely used in economics [38] where service providers employ price variations to ensure supply and demand balance over time. Such techniques are exploited in various applications such as energy consumption optimization in smart grid power supply systems [32, 25] and mobile internet access [6]. Time utility functions were designed to express the temporal demand and reward or penalize the output depending on the desired execution time limit. They are widely used in real time computing [22] for resource management such as package scheduling in wireless networks [5] or video transcoding [35]. In contrast to these works, we are the first to exploit time-based utility functions with different properties in streaming crowdsourcing scenarios.

Finally, a number of works have proposed gamification elements to improve crowdsourcing performance. In [1] various aspects of game mechanics embedded in standard IR tasks are explored, including information seeking, crowdsourcing, and user engagement. He et al. [13] introduce a user interface for studying search behavior within a gamified setting where users receive points for finding relevant documents, and where scores are announced in leaderboards. Dinesh et al. [28] discuss additional incentive structures to encourage user activity within an online platform. In [11] the authors show that gamification in crowdsourcing can improve both worker engagement and effectiveness. In our previous work [30] we applied competition designs to improve cost and time efficiency of crowdsourcing, and we explored a wide range of monetary reward schemes that are inspired by competitions, lotteries, and games of luck. In a follow up work we investigated how team mechanisms can be leveraged for further improving the cost efficiency [31]. In this paper we will use that competition framework as a sub-component. None of works so far considers gamification or competitions in context of streaming crowdsourcing to stimulate the crowd to adapt to demand changes.

To the best of our knowledge, we are the first to leverage competition designs to adapt crowd performance to dynamic changes in demand and utility and to conduct systematic real-world studies of the effects on both time aware quantity and quality of annotations.

3. TEMPORAL CROWDSOURCING SETTING AND COMPETITIONS

In this section we formalize crowdsourcing scenarios with temporal demand characteristics, describe our existing competition-based crowdsourcing framework, and, building on that framework, introduce our approach for controlling the temporal performance of the crowd.

3.1 Temporal Valuation Framework

In time dependent applications that interact with, or incorporate, a crowdsourcing system, qualitatively different forms of temporal considerations may arise (often in combined form): Firstly, the utility of finished tasks might decrease with processing delays; this might, for instance hold for analytics services where up-to-date information is more valuable. Secondly, obtaining an insufficient number of finished tasks up to a certain point in time might result in decreased utility as might be the case for real time statistical analysis or for online algorithms requiring human-labeled training input.

To formalize and quantify these aspects, let S be a set of task sets issued within an overall time frame $T = [t_{start}, t_{end}]$. Each task set $s \in S$ is issued to the crowdsourcing system individually, with a desired number of annotations d_s and a desired completion time t_s. The crowdsourcing system stops issuing and processing tasks in s at time tr_s and responds with n_s annotations for s. For task set s we define the utility value u(s) of $n_s(tr_s)$ finished tasks at response time tr_s as

$$u(s) = \alpha(d_s, n_s(tr_s)) \cdot \beta(tr_s, t_s) \cdot \gamma(s), \tag{1}$$

where the *completion factor* $\alpha(d_s, n_s(tr_s))$ is a utility term depending on extent to which the batch has been annotated, $\beta(tr_s, t_s)$ is a *delay factor* accounting for elapsed time, and $\gamma(s)$ is a *base utility factor* for task set s. Our goal is to maximize the overall utility $U = \sum_{s \in S} u(s)$.

3.2 Utility Scenarios

We now describe various manifestations of the individual utility factors in Equation 1 based on different application scenarios.

3.2.1 Completion Factor

The value of the completion utility factor depends on the number of tasks in a task set that are finished at response time. The concrete form of this factor can vary with the application context. Figure 2 visualizes examples of possible functional representations that we describe and motivate in the following.

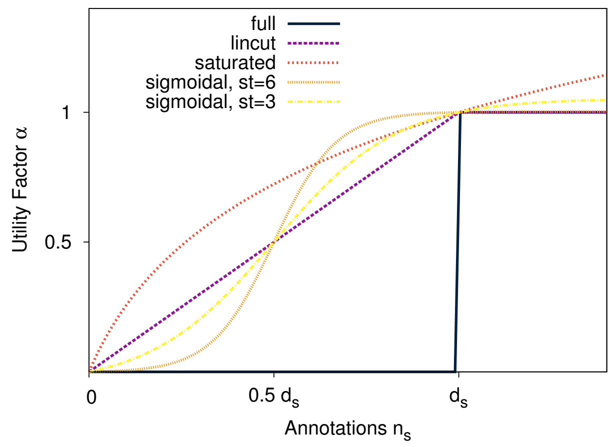


Figure 2: Example of the different completion factors α depending on number of annotations n_s for demand d_s.

Full Completion. As a first basic scenario, we model the completion factor for the case that partial completion does not yield any utility, i.e. each task set has to be fully completed in order to be useful:

$$\alpha_{full}(d_s, n_s(tr_s)) = \begin{cases} 0, & \text{for } n_s(tr_s) < d_s \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

An example are complex tasks which are decomposed into sub-tasks for annotation, such as nutritional analysis based on photos of meals [26].

Linear with Cut-off. We model scenarios where partial completion is acceptable but where a surplus of annotations does not add any additional value, using a linear function with cut-off:

$$\alpha_{lincut}(d_s, n_s(tr_s)) = \frac{\min(n_s(tr_s), d_s)}{d_s}. \tag{3}$$

This function is applicable if a fixed number of tasks needs to be processed with each individual item having a value by itself as, for instance, annotations related to product categorization for online retailers like Amazon[2].

Saturated. Saturation effects can occur in scenarios where the value of additional tasks decreases with the number of already processed tasks. In such a scenario, a monotonically increasing convex utility function is applicable. More specifically, we model utility at time tr_s as a logarithmic growth function:

$$\alpha_{saturated}(d_s, n_s(tr_s)) = \log_b\left(1 + \frac{(b-1) \cdot n_s(tr_s)}{d_s}\right), \tag{4}$$

where parameter b controls the steepness of the incline. This can be regarded as a “softer” variant of the previously described cut-off model. A concrete application scenario is machine learning where the accuracy of the method often improves with training set size in a similar fashion [33, 27]. Another example is the accuracy of crowdsourced labels obtained through redundant annotations, which, modeled as an ensemble of independent classifiers, shows similar characteristics [17].

[2]`https://amazon.com`

Sigmoidal. For machine learning or statistical analysis, there might be situations where a small number available annotations have very limited value, as no sufficiently reliable predictions can be made, whilst saturation effects might occur for larger numbers of annotations. Such scenarios can be addressed with a utility function that is concave at first and then becomes convex as before. More specifically, we model utility at time tr_s as

$$\alpha_{sigmoidal}(d_s, n_s(tr_s)) = sc \cdot \left(\frac{1}{1+e^{st \cdot \; 1-2 \cdot \frac{n_s(tr_s)}{d_s}}} - \frac{1}{1+e^{st}} \right), \tag{5}$$

where *sc* is a scaling factor and *st* determines the steepness of the curve. Such sigmoidal utility functions are also employed in the context of wireless networks [5, 36].

3.2.2 Delay Factor

Our delay factor model is inspired by Jensen's time-utility functions, which were introduced in the context of real-time processing systems [19, 18] and which were also applied to package scheduling in networks [34]. In a scenario with hard time bounds the delay factor can be modeled as

$$\beta_{hard}(tr_s, t_s) = \begin{cases} 1, & \text{for } tr_s \leq t_s \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

i.e. tasks finished after the response time have no value. A soft version of this assigning a diminished value to tasks processed can be described using an exponential decay function as follows:

$$\beta_{soft}(tr_s, t_s) = \begin{cases} 1, & \text{for } tr_s \leq t_s \\ e^{c \cdot \; t_s - tr_s}, & \text{otherwise,} \end{cases} \tag{7}$$

where c determines sharpness of decay, after the desired completion time has passed.

3.2.3 Base Utility

Finally, the base utility factor $\gamma(s)$, is use case specific. In the following we assume that individual tasks within a task set contribute equally to the base utility. Thus the base utility is proportional to the number of desired annotations $\gamma(s) \propto d_s$.

3.3 Competitions and Incentives for Dynamic Crowdsourcing Demands

We now introduce crowdsourcing competitions [30, 31], a crowdsourcing scheme that is attractive due to its low cost per annotation compared to standard pay-per-task micro-task crowdsourcing. As an extension, we introduce incentive mechanisms for dynamic processing demands.

We consider a scenario with a fixed monetary budget M for paying workers and time frame T, as defined in Section 3.1, and a crowd consisting of n workers $W = w_1, .., w_n$. This budget is distributed among the (individual and teams of) workers in a team-based crowdsourcing competition, depending on the values produced by them. A worker w_i receives a monetary reward $m(w_i)$ with $\sum m(w_i) = M$.

Individual Competitions. In individual competitions, workers are ranked according to their produced values and the reward of a worker depends on his or her rank. Formally, let $\mathit{rank}(w_i) \in \{1, \ldots, n\}$ be the rank of worker w_i, with a rank of j corresponding to the j-th highest value produced across all workers. The individual reward $m_{ind}(w_i)$ is computed as a monotonically decreasing, convex function $\Gamma(\mathit{rank}(w_i))$ of the worker's rank, i.e. top performers receive more money per solved task.

Team-based Competitions. In addition, we rank teams of workers according to the sum of the values produced by their members, with $\mathit{rank}(t_i)$ defined analogous to the individual ranks. A team reward $mr(t_i)$ is assigned to each team, using, as for individual competitions, a monotonically decreasing, convex function. A worker w_j receives a reward $m_{team}(w_j)$ that is proportional to the contribution to his or her team $t(w_j)$ in addition to his or her individual reward $m_{ind}(w_j)$. The teams are formed through a "balanced" strategy for team formation, where we set up a fixed number τ of empty teams and assign each new worker (from the stream of registering workers) uniformly at random to one of the teams containing the lowest number of workers. Furthermore, in order to avoid large numbers of inactive team members, we require a low initial threshold for the value produced by worker w_i before assigning him or her to one of the teams.

Information on Competitors. We employ a *medium* information policy where only part of the information about other workers and teams is revealed: workers are shown their scores and rank along with their k neighbors above and below in the leaderboard. Additionally, the team leaderboard also keeps workers updated about their current shares of the team reward.

Quality Check and Feedback Mechanism. A "honeypot" task drawn from the gold standard is randomly introduced within each batch of *bs* tasks. After workers finish a batch, they are shown the honeypot and their input for it. If workers solved the honeypot correctly, a batch reward *br* is added to their score, otherwise a penalty *bp* $\propto$ *br* is subtracted instead (with a cut-off threshold of 0 for the score, i.e. there are no negative scores).

Incentives for Dynamic Processing Demands. Our objective is to influence aggregate crowd behavior towards desired temporal annotation output characteristics. To this end, at each point in time $t \in T$, a correct task annotation is rewarded with $r(t)$, i.e. the batch reward is $br(t) = r(t) \cdot bs$. The reward per task at time t is announced at time $a(t) \leq t$ to the workers.

In our experiments we consider the case of fixed-length periods of increased annotation demand, which we match with periods of increased reward rates framed as "bonus events". During those bonus events, the reward rate r is increased by a constant bonus factor B in comparison to a constant base reward rate ($r = 1$ in our case). In addition, we assume there is fixed forecast period fp for all tasks, i.e. $a(t) = t - fp$, for all $t \in T$. As an extension of this mechanism, we further introduce the notion of "limited bonus", where the bonus is granted only for a limited number l of annotations. However our general framework covers many alternative scenarios, involving, for instance, variations in forecast and bonus periods, and dynamic adjustment of bonus factors within the competition and within bonus events.

4. EXPERIMENTS

In this section we evaluate the demand-based control strategies described in Section 3. The objective of our evaluation was to study the temporal adaptation of the crowd under different demand scenarios and incentives, and to evaluate the effectiveness of our approach in achieving more desirable distributions of work throughput over time.

4.1 Setup

Crowdsourcing Task. We launched a *face recognition* task employed in our previous work [31], where workers were asked to identify a person on a given reference photo among a set of 10 test photos. The images were retrieved from the PubFig[3] database which was created for face verification [21]. Out of originally 58,797 images with faces of 200 celebrities, 37,004 images were available on the Web. We reviewed the dataset manually and removed 663 images showing placeholders as well as 135 images we deemed unsuitable because the correct person was not shown on the image.

Implementation and Settings. For each reward scenario we conducted one competition and announced a corresponding crowdsourcing task on the CrowdFlower platform and a mailing list consisting of participants from the previous competition about one day before the competition started. The workers were choosing the tasks autonomously, as common for crowdsourcing platforms such as CrowdFlower. As the tested reward strategies are not supported on that crowdsourcing platform, we ran the actual competition using an external application on servers at our institute. Each worker was assigned a user code upon registration which had to be submitted to the CrowdFlower task in order for the account to be activated, thus ensuring that each participant could be paid. All experiments were performed with a duration of three days and a break of one day in between two competitions to avoid extensive user fatigue.

Demand Scenario. In our scenarios we considered the use case that task sets are issued sequentially at the beginning of each hour with a response time of one hour, and that for each hour there exist demands d_s for the completion of tasks. We tested two forecast scenarios: forecast period $fp = 1$ hour (short notice) and $fp \geq \max(T) - \min(T)$ (long notice - demand is known beforehand). For all hours of the competition, demands d_s were set to a constant base demand of d_{base} annotations - except for 10 randomly chosen "peak" hours where the demand was doubled. This corresponds to the important case that higher annotation throughput should be obtained in certain time intervals involving increased demand.

Tested Strategies. In our previous works on crowdsourcing competitions [30, 31] we observed bursts in participation for the first few competitions launched; participation numbers then became more stable for later runs. To reduce these novelty effects and to ensure better comparability of our experiments, we started with three preliminary experimental rounds consisting of a competition without demand changes and two competitions with peak hours announced as schedules in advance.

[3]`http://www.cs.columbia.edu/CAVE/databases/pubfig/`

In our main experiments, we increased the reward during bonus events by different factors B: double points per batch (**low**), five times points per batch (**mid**), and ten times points per batch (**high**). Each setting was tested under two forecast periods fp as described in the previous paragraph, corresponding to different time intervals for advance notification of bonus events to participants: In the **long** notice setting, participants had access to the full event schedule already before the competition started. In the **short** notice setting, events were announced one hour beforehand. As baseline we conducted a team-based crowdsourcing competition with balanced team strategy and individual rewards as in [31] without any additional incentives during peak hours (**baseline**). The resulting 7 individual experimental runs were carried out in randomized order.

To address problem settings where more annotations might not offer additional value at a certain point, we conducted an additional experiment with a limited bonus pool mechanism (**limited**): during peak hours, bonus points were only awarded for the first l annotations. Based on results from our main experiments we chose limit $l = 6000$. For notice and bonus settings, we chose **long** notice and **high** bonus in expectation of more clear results. In all experiments, we distributed a prize money of $M = 100$ USD among the top-5 performing teams and individuals.

4.2 Results

In our experiments, overall 2.36 million images were matched correctly by 921 participants from 76 different countries (amounting to over 6200 hours of work). The workers were accessing our competitions from several parts of the world, with a majority from Europe (34.9%) and South America (27.6%). The country represented most numerously was Venezuela (17.9%), followed by India (9.1%) and the United States (5.8%).

4.2.1 Aggregate Results

Annotation Throughput. Table 1 shows the number of correctly annotated images for each of the experiments, along with the peak-to-non-peak ratio of mean annotation throughput per hour, overall number of participants, and peak-to-non-peak ratio of average number of annotators per hour. The main results are the following:

- As expected, for the *baseline* experiment the mean throughput in peak hours does not deviate noticeably from throughput in non-peak hours, as no bonus mechanism was employed. Differences are due to randomness in the distribution of peak demands and uneven base annotation throughput distribution in time (see Figure 4g).
- For the *short notice* experiments, *medium bonus* increases peak-to-non-peak ratio of mean annotation throughput in comparison to *low bonus*, while increasing the bonus further did not offer additional benefits in this setting.
- In contrast, in the *long notice* setting further substantial increase of the throughput peak-to-non-peak ratio can be observed when *high bonus* is employed.

Experiment	Correct Annotations	Mean Throughput	Annotators	Mean no. of Annotators
		peak-to-non-peak ratio	Overall	peak-to-non-peak ratio
Baseline				
no bonus	316,769	1.02	208	1.14
Short Notice				
low bonus	271,963	2.42	164	1.41
medium bonus	272,549	3.02	214	1.46
high bonus	319,236	2.96	260	1.62
Long Notice				
low bonus	376,828	2.34	291	1.55
medium bonus	284,596	3.26	179	1.76
high bonus	247,901	4.21	167	1.95

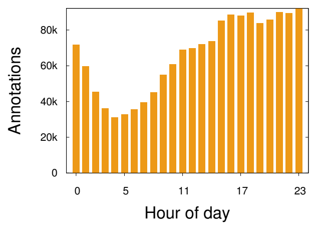
Table 1: Aggregate outcomes of the experimental rounds.

Statistical Tests. We validated our results against false positive findings that may occur as a result of varying base annotation behavior, experiment order, or peak demand distribution. To this end, we employed Welch's unequal variances t-test for comparing the average number of annotations during peak and non-peak hours. For all experiments involving bonus incentives, the differences between average peak and non-peak were significant with $p < 0.001$.

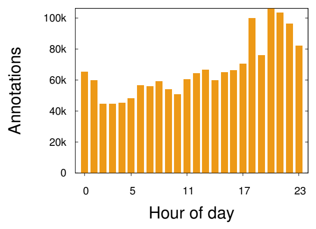
Participation. The number of annotators given in Table 1 provides further insight into differences between short notice and long notice settings. The ratio of average number of annotators per hour for peak and non-peak hours varies drastically for the different settings. For both short and long notice, the ratio increases with the bonus amount; however, under all bonus settings, the ratio is higher for long notice. This is not surprising, as participants have advance knowledge of bonus events in the long notice setting and can plan accordingly, while in the short notice setting participants have to monitor the application for presence of a bonus. In combination, we find that increasing bonus can increase peak throughput significantly but is limited in the number of workers that can be mobilized. This indicates that advance notice, based on forecasts for instance, is necessary to match more extreme peaks in annotation demand.

Annotation Quality. Similar to previous results [31], the large majority of images was annotated correctly, with overall accuracy ranging from 93.9% in the long notice, medium bonus experiment, to 96.4% in the long notice, low bonus experiment. In comparison to the baseline, with overall accuracy of 95.0%, there is no noticeable difference. In addition, there is no difference in annotation accuracy for peak hours compared to non-peak hours in our experiments.

Base Offer Distribution. Figure 3 shows the distribution of annotations per hour of day (in hours where no bonus incentive was applied) from the workers perspective in their local time, as well as from our perspective in CEST, aggregated over all workers in all experiments[4]. The distribution



(a) Worker local time



(b) CEST

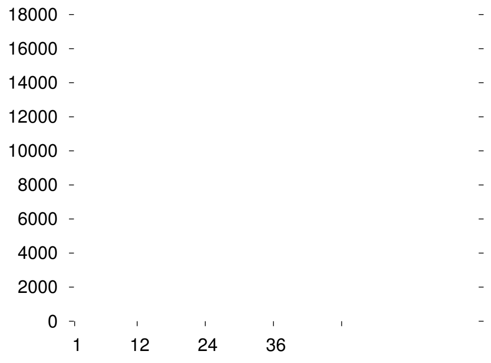
Figure 3: Non-peak annotations per hour of day for a) worker local time and b) CEST aggregated for all experiments.

according to the workers' local time in Figure 3a shows pronounced differences based on time of day, with a general preference for annotating in the evening. Surprisingly, even in late night hours there is substantial activity. From the global point of view (Figure 3b), time zone effects interfere with each other and differences are less pronounced; however, variations in annotation output depending on time of day are still clearly visible. In addition, we can observe two peaks in annotation output featured in the CEST plot. There is a peak between 6pm and 7pm, which possibly occurs due to our experimental schedule: experiments started and ended at 6pm CEST. There are further differe hos depending on the location of workers. The socond peak, between 8pm and 10pm CEST, aligns with the time of greatest activity of European workers, who contributed 38% towards the annotations obtained in that time frame, compared to a contribution of 34% in total.

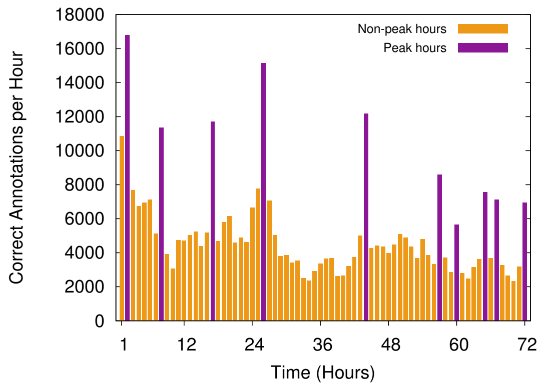
4.2.2 Temporal Annotation Characteristics

Annotations in Peak and Non-Peak Hours. Figure 4 shows number of correct annotations per hour for all experiments. We distinguish between annotations obtained in peak demand hours and annotations obtained in hours without peak demand. For the baseline, in Figure 4g, we can see that peak demand had no influence on annotation throughput; differences in annotation throughput in these

[4]Time zone information was obtained from the workers' browsers and depends on their system settings.



(a) short notice, low bonus



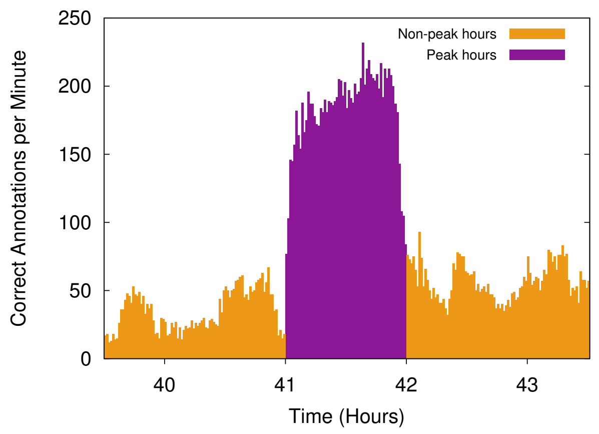


Figure 5: Number of correct annotations per minute around a typical bonus hour.

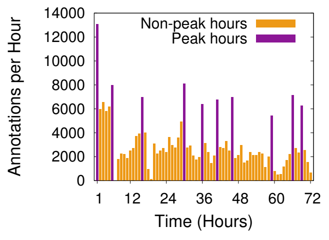
hours align with the overall progression of measurements. In addition, the peak demand hours fell mostly into the first half of the experiment, influencing the annotation throughput and number of annotators peak-to-non-peak ratios observed before.

In contrast, the plots for all variations of our bonus method show clearly increased annotation throughput for peak demands[5]. Furthermore, Figure 4f shows that the combination of high reward and long notice results in extreme annotation peaks in response to the demand peaks, compared to a rather low base annotation throughput due to lower participation numbers (see Table 1). In all bonus settings, the magnitude of peak hour annotation throughput varies in a similar fashion to base annotation throughput in non-peak hours, depending on factors such as hour of day and general activity in the competition.

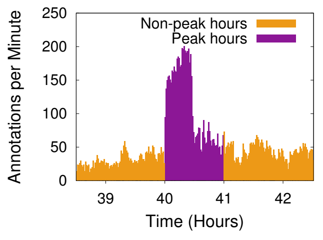
Temporal Variations around Bonuses. A more fine-grained view of annotation throughput can be seen in Figure 5 showing annotations per minute for a period of 1.5 hours before and after a bonus hour in the long notice, medium bonus experiment, as a typical example. Most notably, we observe a drop shortly before the bonus is applied. This “anticipation period” is a pattern that can be observed for most bonus hours.

In the beginning of bonus hours increased throughput levels are reached in a sharp incline after a short delay; compared to annotation rate in the preceding hour, throughput is increased by at least 50% in less than two minutes on average. On the other hand, towards the end of bonus hours a decline can be observed. As a result of our quality mechanism, scores are granted only upon completion of a batch of 100 images. A participant who estimates he or she will not complete the batch in time before the end of the bonus hour may not expect to receive bonus points for the annotations. These observations can be taken into account when planning the bonus based on a demand scenario: our demand scenario could be matched more closely by awarding the bonus slightly earlier and longer.

[5]Except for one hour in Figure 4f. Activity during bonus hours was so high in the high bonus setting, in fact, that our application could not handle the load at some points, which resulted in artifacts in the data.



(a) Annotations per hour



(b) Annotations per minute

Figure 6: Peak and non-peak annotations per hour (a) and annotations per minute around a typical bonus hour (b) for the limited bonus experiment.

4.2.3 *Limited Bonus*

We also studied a complementary mechanism to control the effect of our bonus incentive scheme with more precision. To this end, for the high bonus - long notice setting, we introduced and announced to workers a limit on the number of annotations that were awarded a bonus upon completion. Figure 6a shows the number of correct annotations per hour over the course of the experiment[6]. Except for the first bonus hour at the very beginning of the competition, for the most part annotation numbers for bonus hours are closely above our set limit of $l = 6000$ processed tasks, and exhibit less variation for those hours in comparison to the setting without limit. A more fine-grained view shown in Figure 6b for the example of bonus hour 6 reveals throughput levels comparable to peak hours in the previous experiments as long as the bonus is available which afterwards drops approximately to the level of non-bonus hours. These results illustrate the viability of our method for further adjusting temporal crowd performance to annotation demands.

4.2.4 *Utility Scenarios*

We further evaluated our methods by comparing overall Utility U, as defined in Eq. 1, in Table 2 for several variations on completion factors α, parameterized with $b = 8$, $sc = 1$, and $st = 3$ (controlling the steepness of saturated completion factor, scaling of sigmoidal utility factor, and steepness of sigmoidal utility factor, respectively). We employed hard time constraints β_{hard}, base utility $\gamma = s$ for non-peak hours and an increased base utility $\gamma = 10 \cdot s$ for peak-hours, in accordance with our scenario, focusing on matching peak annotation demands. To eliminate influences on absolute annotation numbers, such as the order of experiments, we normalized the annotations n_s as well as the demand d_s by their respective sums for each of the experiments.

For the described setting, all of our methods outperform the baseline with respect to utility. This is because peak-hour annotation demands, which are of high interest in our setting, are well matched (cf. Figure 4). For completion factor $\alpha_{saturated}$, annotation output in the long notice, high bonus experiment results in highest overall utility, in spite of the fact that the observed effect was extreme (peak-to-non-peak throughput ratio of 4.2), at the expense of com-

[6]Note that in hours 7, 18, and 19 of the competition unfortunately there occurred application failures, resulting in strongly reduced throughput for those hours.

Experiment	Utility U		
	α_{lincut}	$\alpha_{saturated}$	$\alpha_{sigmoidal}$
Baseline			
no bonus	2.1	2.62	2.12
Short Notice			
low bonus	2.96	3.27	3.09
medium bonus	2.91	3.38	3.05
high bonus	2.98	3.40	3.12
Long Notice			
low bonus	2.81	3.23	3.01
medium bonus	3.02	3.47	3.15
high bonus	2.83	3.55	2.95
limited bonus	3.01	3.39	3.13

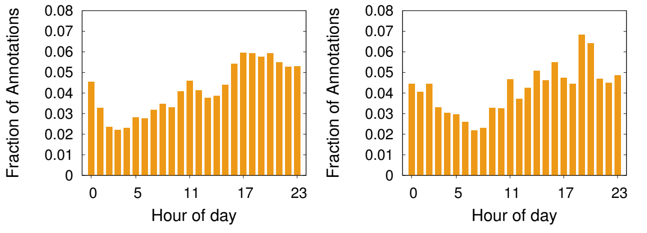
Table 2: Overall utility U under several completion factor scenarios with increased base utility factor during peak-hours and hard time bounds delay factor.

pletion rates during non-peak hours. In contrast, sigmoidal $\alpha_{sigmoidal}$ and linear cut-off $\alpha\, lincut$ completion factors penalize low completion rates more harshly and lead to a relatively lower overall utility. In these settings, annotation output in the long notice, medium bonus experiment, with a more moderate peak-to-non-peak throughput ratio of 3.3, achieved highest valuation. Similarly, utility of annotation results in the limited bonus experiment (as an extension on the long notice, high bonus setting) is comparable, demonstrating the usefulness of the mechanism.

4.2.5 *Influences on Worker and Competition Dynamics*

In the following, we examine possible effects of bonus mechanics on worker behavior and on competition dynamics. Some participants might use the setting to their advantage, whilst others are discouraged. In addition, the setting might put additional strain on the workers.

Distribution of Annotations throughout the Day. Figure 7 shows the distribution of annotations over hours of day based on individual participant time zones aggregated over all users for the baseline and long notice, medium bonus experiments. The distribution observed for the baseline experiment does not differ from the aggregate results shown in Figure 3. In bonus settings however, as shown for long notice, medium bonus as an example in Figure 7b, bonus hours result in spikes that deviate from the normal daily annotation pattern of the workers. Although occurrence of bonus hours in relation to worker local time is random, we observe a common pattern of increases in annotation output occurring predominantly in the afternoon and evening hours. This indicates that our method amplifies time zone effects in annotation output of the users; however, as the bonus hours match our demand scenario, the effects counteract in general in a useful way. In addition, we do not see a general pattern of changes in the day-night cycles of participants, which may have adverse effects on them.

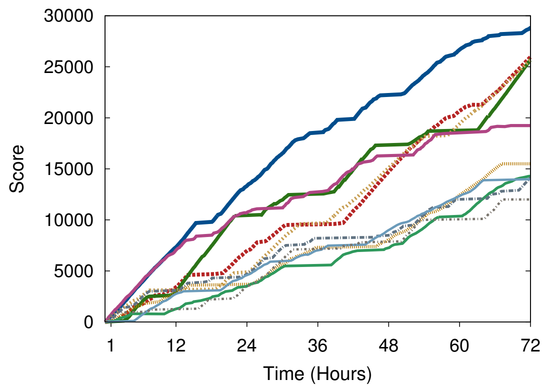


(a) Baseline (b) long notice, medium bonus

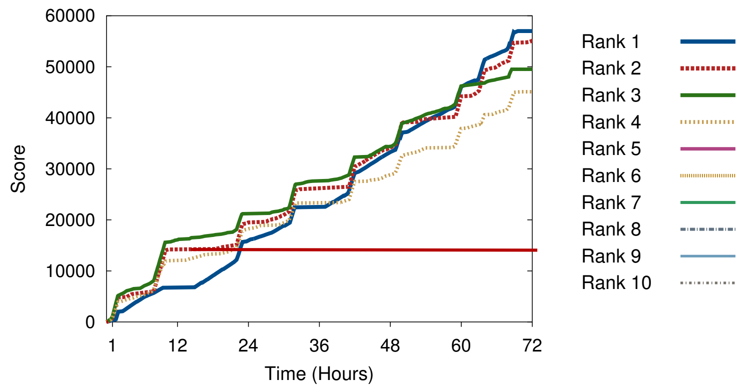
Figure 7: Distribution of annotations per hours of day, based on the participants' time zones for the baseline and long notice, medium bonus experiments.

Score Developments during Experiments. As an example for general patterns that can be observed, Figure 8 shows the scores over time for the top 10 individuals and teams in the baseline and long notice, medium bonus experiments. For bonus settings, we observed jumps in the scores during bonus hours. The effect is more pronounced, the higher the bonus. The bonus mechanics allow individuals and teams to radically improve their score in a short time, which might have an adverse effect on the competition setting. Especially in high bonus settings, missing out on only a small numbers]TJ T [(o)

next bonus I will email you. ". This behavior shows strong advantages of our gamification-based methods in dealing with unexpected events, which would be much more of a challenge in a classic pay-per-task crowdsourcing scenario.



(a) Baseline individual scores



(b) Long notice, medium bonus individual scores

Figure 8: Scores over time for the top 10 individual participants in the baseline and long notice, medium bonus experiments.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have studied the problem of stream alike data processing in crowdsourcing scenarios. To this end, we proposed a temporal valuation and utility framework as well as incentive schemes to control crowd performance as would be required in many real world stream processing applications. Our evaluation shows the viability of our techniques for flexibly adjusting worker output according to demand changes over time with a fixed budget and considerable improvement of the time-adjusted value produced in various utility scenarios when compared to the baseline. In particular, for peak demand periods, our most effective strategy increases the crowd throughput by more than 300% on average.

In our future work we aim to extend our techniques for effecting desired throughput distributions to accommodate a variety of demand scenarios defined in our temporal valuation framework. To this end, we plan to involve crowd performance forecasting techniques to support adjustment of the input parameters on the fly and to carry out in-depth studies on the effects of dynamic parameter changes.

6. ACKNOWLEDGMENTS

This work is partly funded by the European Research Council under ALEXANDRIA (ERC 339233), by the European Commission under grant agreements 619525 (QualiMaster) and 688135 (STARS4ALL).

7. REFERENCES

[1] *GamifIR '14: Proceedings of the First International Workshop on Gami cation for Information Retrieval*, 2014. ACM.

[2] M. S. Bernstein, J. Brandt, R. C. Miller, and D. R. Karger. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 33–42. ACM, 2011.

[3] A. Biem, H. Feng, A. V. Riabov, and D. S. Turaga. Real-time analysis and management of big time-series data. *IBM J. Res. Dev.*, 57(3-4):1:8–1:8, May 2013.

[4] J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, B. White, S. White, et al. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 333–342, 2010. ACM.

[5] S. Dahi and S. Tabbane. Sigmoid utility function formulation for handoff reducing access model in cognitive radio. In *Communications and Information Technologies (ISCIT), 2013 13th International Symposium on*, pages 166–170, 2013.

[6] A. Damasceno, R. A. Mini, J. M. Almeida, and H. Marques-Neto. A base station workload-aware dynamic pricing scheme for mobile internet access. In *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*, MobiArch '15, pages 45–50, 2015. ACM.

[7] A. Das Sarma, A. Parameswaran, H. Garcia-Molina, and A. Halevy. Crowd-powered find algorithms. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 964–975. 2014, IEEE.

[8] D. E. Difallah, M. Catasta, G. Demartini, and P. Cudré-Mauroux. Scaling-up the crowd: Micro-task pricing schemes for worker retention and latency improvement. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[9] J. Fan, M. Zhang, S. Kok, M. Lu, and B. C. Ooi. Crowdop: Query optimization for declarative crowdsourcing systems. *Knowledge and Data Engineering, IEEE Transactions on*, 27(8):2078–2092, Aug 2015.

[10] S. Faradani, B. Hartmann, and P. G. Ipeirotis. What's the right price? pricing tasks for finishing on time. *Human computation*, 11, 2011.

[11] O. Feyisetan, E. Simperl, M. V. Kleek, and N. Shadbolt. Improving paid microtasks through gamification and adaptive furtherance incentives. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 333–343, 2015. International World Wide Web Conferences Steering Committee.

[12] Y. Gao and A. Parameswaran. Finish them!: Pricing algorithms for human computation. *Proceedings of the VLDB Endowment*, 7(14):1965–1976, 2014.

[13] J. He, M. Bron, L. Azzopardi, and A. de Vries. Studying user browsing behavior through gamified search tasks. In *Proceedings of the First International Workshop on Gami cation for Information Retrieval*, GamifIR '14, pages 49–52, 2014. ACM.

[14] M. Imran, C. Castillo, J. Lucas, P. Meier, and S. Vieweg. Aidr: Artificial intelligence for disaster response. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 159–162, 2014. International World Wide Web Conferences Steering Committee.

[15] M. Imran, C. Castillo, J. Lucas, M. Patrick, and J. Rogstadius. Coordinating human and machine intelligence to classify microblog communications in crises. *Proc. of ISCRAM*, 2014.

[16] M. Imran, I. Lykourentzou, Y. Naudet, and C. Castillo. Engineering crowdsourced stream processing systems. *arXiv preprint arXiv:1310.5463*, 2013.

[17] P. G. Ipeirotis and P. K. Paritosh. Managing crowdsourced human computation: a tutorial. In *Proceedings of the 20th international conference companion on World wide web*, pages 287–288. ACM, 2011.

[18] E. D. Jensen. Asynchronous decentralized realtime computer systems. In *Real Time Computing*, pages 347–371. Springer, 1994.

[19] E. D. Jensen, C. D. Locke, and H. Tokuda. A time-driven scheduling model for real-time operating systems. In *RTSS*, volume 85, pages 112–122, 1985.

[20] T. Johnson, S. Muthukrishnan, and I. Rozenbaum. Sampling algorithms in a stream operator. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 1–12, 2005. ACM.

[21] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and Simile Classifiers for Face Verification. In *IEEE International Conference on Computer Vision (ICCV)*, Oct 2009.

[22] P. Li, B. Ravindran, and E. Jensen. Adaptive time-critical resource management using time/utility functions: past, present, and future. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, volume 2, pages 12–13 vol.2, Sept 2004.

[23] W. Mason and D. J. Watts. Financial incentives and the "performance of crowds". *SIGKDD Explor. Newsl.*, 11(2):100–108, May 2010.

[24] E. Minack, W. Siberski, and W. Nejdl. Incremental diversification for very large sets: A streaming-based approach. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 585–594, 2011. ACM.

[25] A.-H. Mohsenian-Rad and A. Leon-Garcia. Optimal residential load control with price prediction in real-time electricity pricing environments. *Smart Grid, IEEE Transactions on*, 1(2):120–133, Sept 2010.

[26] J. Noronha, E. Hysen, H. Zhang, and K. Z. Gajos. Platemate: crowdsourcing nutritional analysis from food photographs. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 1–12. 2011, ACM.

[27] C. Perlich, F. Provost, and J. S. Simonoff. Tree induction vs. logistic regression: A learning-curve analysis. *The Journal of Machine Learning Research*, 4:211–255, 2003.

[28] D. Pothineni, P. Mishra, A. Rasheed, and D. Sundararajan. Incentive design to mould online behavior: A game mechanics perspective. In *Proceedings of the First International Workshop on Gami cation for Information Retrieval*, GamifIR '14, pages 27–32, 2014. ACM.

[29] S. Roche, E. Propeck-Zimmermann, and B. Mericskay. Geoweb and crisis management: issues and perspectives of volunteered geographic information. *GeoJournal*, 78(1):21–40, 2013.

[30] M. Rokicki, S. Chelaru, S. Zerr, and S. Siersdorfer. Competitive game designs for improving the cost effectiveness of crowdsourcing. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, CIKM '14, pages 1469–1478, 2014. ACM.

[31] M. Rokicki, S. Zerr, and S. Siersdorfer. Groupsourcing: Team competition designs for crowdsourcing. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 906–915, 2015. International World Wide Web Conferences Steering Committee.

[32] P. Samadi, A.-H. Mohsenian-Rad, R. Schober, V. Wong, and J. Jatskevich. Optimal real-time pricing algorithm based on utility maximization for smart grid. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 415–420, Oct 2010.

[33] D. R. Stockwell and A. T. Peterson. Effects of sample size on accuracy of species distribution models. *Ecological modelling*, 148(1):1–13, 2002.

[34] J. Wang and B. Ravindran. Time-utility function-driven switched ethernet: Packet scheduling algorithm, implementation, and feasibility analysis. *Parallel and Distributed Systems, IEEE Transactions on*, 15(2):119–133, 2004.

[35] Y. Wang, J.-G. Kim, and S.-F. Chang. Content-based utility function prediction for real-time mpeg-4 video transcoding. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I–189–92 vol.1, Sept 2003.

[36] M. Xiao, N. B. Shroff, and E. K. Chong. A utility-baed power-control scheme in wireless cellular systems. *Networking, IEEE/ACM Transactions on*, 11(2):210–221, 2003.

[37] D. Yang, E. A. Rundensteiner, and M. O. Ward. Summarization and matching of density-based clusters in streaming environments. *Proc. VLDB Endow.*, 5(2):121–132, Oct. 2011.

[38] S. Yousefi, M. P. Moghaddam, and V. J. Majd. Optimal real time pricing in an agent-based retail market using a comprehensive demand response model. *Energy*, 36(9):5716–5727, 2011.