HOCHSCHULE **TRIER**
Trier University of Applied Sciences
Informatik - Computer Science

Informatik-Bericht Nr. 2016-1

Schriftenreihe Fachbereich Informatik, Hochschule Trier

It's about time: Online Macrotask Sequencing in Expert Crowdsourcing

Heinz Schmitz
Trier University of Applied Sciences, Germany
h.schmitz@hochschule-trier.de
Ioanna Lykourentzou
Luxembourg Institute of Science and Technology
ioanna.lykourentzou@list.lu

January 15, 2016

Abstract

We introduce the problem of Task Assignment and Sequencing (TAS), which adds the timeline perspective to expert crowdsourcing optimization. Expert crowdsourcing involves macrotasks, like document writing, product design, or web development, which take more time than typical binary microtasks, require expert skills, assume varying degrees of knowledge over a topic, and require crowd workers to build on each other's contributions. Current works usually assume offline optimization models, which consider worker and task arrivals known and do not take into account the element of time. Realistically however, time is critical: tasks have deadlines, expert workers are available only at specific time slots, and worker/task arrivals are not known a-priori. Our work is the first to address the problem of optimal task sequencing for online, heterogeneous, time-constrained macrotasks. We propose TAS-ONLINE, an online algorithm that aims to complete as many tasks as possible within budget, required quality and a given timeline, without future input information regarding job release dates or worker availabilities. Results, comparing TAS-ONLINE to four typical benchmarks, show that it achieves more completed jobs, lower flow times and higher job quality. This work has practical implications for improving the Quality of Service of current crowdsourcing platforms, allowing them to offer cost, quality and time improvements for expert tasks.

1 Introduction

As the appeal of crowd work increases, there is a growing need to provide support for more complex tasks and workflows. Examples of such tasks include document editing, product design, social innovation and idea development, offered through dedicated platforms (upWorks[1], crowdSpring[2] etc.), or incorporated into traditional ones through complex workflows (e.g. the recently launched CrowdFlower Labs[3]). This type of crowdsourcing is often referred to as *expert crowdsourcing*, and the tasks that it involves are referred to as *macrotasks* [15]. Macrotasks differ from the typically crowdsourced micro-

[1] https://www.upwork.com/
[2] http://www.crowdspring.com/
[3] http://www.crowdflower.com/blog/introducing-crowdflower-labs

tasks in that they require expert skills, assume varying degrees of knowledge over a topic, may take more worker time and often involve task dependency, i.e. workers building on each other's contributions.

Together with the demand for complex tasks and their supporting workflows, customers are increasingly interested in performance guarantees, i.e. the *optimization* of expert crowdsourcing in terms of cost, quality and timeliness. Recent studies that examine expert crowdsourcing optimization [2, 14] typically seek to find worker assignments per task such that the worker contributions add up to a required quality threshold within a given budget. Roughly speaking, the crowdsourced tasks play the roles of multiple knapsacks with some additional concepts like domain-specific expertise and wages per worker, different models of acceptance probabilities or types of quality aggregation. Unfortunately current studies do not take into account the aspect of time, for example in terms of task deadlines, worker time constraints, or time-dependent worker/task variability. As such these studies examine only the *assignment part* of the worker allocation problem (finding which worker should take which task), but not the *sequencing part* (identifying when each worker should contribute). Moreover they usually assume an offline setting, where the algorithms are provided with the complete worker/task input information at once. *In a realistic crowdsourcing setting however, time is an inherent property*: customers require the tasks to finish upon a certain deadline, expert workers are available only at specific time slots, and worker/task arrival or departure information is not a-priori known. Optimizing for time is thus crucial, and raises the need not only for worker-to-task assignment but also for sequencing. It also raises the need for online rather than index-based algorithms, which can take efficient sequencing decisions having access only to time-dependent information that is available until their decision point.

In this paper we introduce the problem of crowdsourcing Task Assignment and Sequencing (TAS), which adds the *timeline perspective* to the crowdsourcing allocation optimization model: How can we find task assignments that can be rolled out in a realistic timeline, featuring unknown task release dates and worker availabilities, as it is the case for real platforms?

To the best of our knowledge, this is the first work that addresses the problem of assignment *and* sequencing optimization for expert crowdsourcing tasks. Overall, our three main contributions with this paper are:

- We explicitly add the *timeline perspective* to task assignment modeling in expert crowdsourcing. That is, our models include not only the worker-to-task-assignments, but also the rolling out of these assignments along a timeline under reasonable constraints. We call this problem modeling TAS and prove its strong NP hardness.

- We propose a *online algorithm*, TAS-ONLINE, which seeks to complete as many jobs as possible within budget, required quality and given timeline, by computing worker sequence-to-task matchings, and without any future input information regarding job release dates or worker availabilities.

- We illustrate, through simulated and real-world experiments, that TAS-ONLINE can achieve more completed jobs, lower flow times and higher quality compared to four typical benchmarks.

The rest of this paper is organized as follows. In section 2 we recapitulate the related literature on crowdsourcing optimization, starting from earlier works that focus on micro-tasks and reaching latest research efforts on knowledge-intensive macro-tasks. In section 3 we describe the characteristics of the expert crowdsourcing setting that this work applies and illustrate, through an example, why taking time into account matters in this particular setting. In this section we also formally model the TAS problem and prove its strong NP-hardness. Next, in section 4 we describe the proposed online algorithm (TAS-ONLINE) for the solution of the TAS problem. In section 5, we present and discuss the experimental results, obtained on both simulated and real-world data. These results compare TAS-ONLINE with four benchmarks found in the literature, and show that TAS-ONLINE achieves higher numbers of completed jobs (both in

terms of absolute value and as a percentage of the solution's upper bound), lower flow times (the time between a job's release date and the latest assignment on that job), better budget utilization and higher levels of quality, comparable only the respective TAS-OFFLINE version for certain of the above measures. Finally, we discuss possible extensions of the TAS model and algorithm (section 6) and end with the paper's main findings and conclusions (section 7).

2 Related Work

2.1 Crowdsourcing Optimization

Crowdsourcing optimization is a term used in various problem settings, including optimizing the selection of worker labor channels to improve performance [22], discovering the optimal worker wage [17], determining the optimal number of workers to undertake each task so as to maximize quality and minimize cost (a method referred to as "plurality optimization" and applicable on n-ary tasks with an objective "true value") [34], or identifying the optimal set of tasks to forward to the crowd (for systems like database query execution ones, which are based partially on crowdsourcing and partially on automated methods) [12].

The family of optimization problems that our work falls into is *allocation optimization*, i.e. the identification of which worker should work on which task and when, in order to optimize one or more global performance metrics, which usually include cost, quality and number of acceptable tasks. This family of problems consists of two distinct optimization problems, *task assignment* and *task sequencing*. Task assignment examines which worker should be given which task. Task sequencing adds the element of time, examining *when* will the worker be given the task. Most existing works focus on the first problem, i.e. task assignment, either for microtasks or for macrotasks. *Microtasks* are tasks that require a small amount of worker time, accept binary (true/false) or n-ary (multiple choice) worker inputs, and the quality of which is determined through methods such as majority voting (assigning multiple workers per microtask). *Macrotasks* [15] require more worker time, accept open-ended continuous (rather than binary or n-ary) worker inputs and their quality is determined through external subjective evaluation (for example peer review).

2.2 Optimizing Task Assignments

Regarding *microtask assignment optimization*, Karger et al. [23] work with homogeneous microtasks (that all have the same level of difficulty and do no distinguish among task "topics"), and propose a matching algorithm inspired by the standard belief propagation algorithm for approximating max marginals, which is order-optimal and minimizes cost. This study is among the first to show that the problem of task matching in crowdsourcing can be transformed to a bipartite graph design problem, where workers are one part of the graph, tasks are the other and the edges represent assignments of workers to tasks. Ho et al. [16] work with heterogeneous microtasks of n-ary classification quality on a model where worker skills per microtask are a-priori unknown, and propose a two phase exploration-exploitation assignment algorithm that seeks to maximize the total benefit of the requester and is competitive with respect to its counterpart of known worker skills. Yuen et al. [43, 44, 42] propose a matrix factorization approach that utilizes the workers' task performance and search history to derive their preferences and perform an improved task-to-worker matching. Regarding *macrotask assignment optimization* Goel et al. [14] and Roy et al. [2, 39] both propose task-to-worker assignment optimizations (the first using a mechanism design-based approach and the second through an index-based approach) on models that consider heterogeneous macrotasks and where the optimization goal is to maximize the utility of the requester while ensuring budget feasibility. Yue et al. [41] add to this model the element of team instead of individual worker assignments, and propose a heuristic genetic algorithm that optimizes for task budget and quality, taking into account worker pay expectations, skills and availability. Jabbari et al. [20] add another interesting facet to the heterogeneous task assignment model, by extending it to cover the online aspect of the problem (workers arrive online, no

prior knowledge over arrivals), and they impose certain constraints that must be respected, such as declaring feasible tasks that workers can handle and the payment they require. The difference of the above works with ours is that their models do not consider the element of time, i.e., they focus only on the task assignment and not the task sequencing aspect of the problem.

2.3 Optimizing Task Timing

Finally, a few studies focus precisely on *time-sensitive optimization*. Regarding *time-sensitive microtasks*, Yu et al. [40] optimize the number of tasks to recommend to each worker per time unit with the objective of maximizing the time average number of successful (i.e. of acceptable quality) jobs for a given time period. Their model assumes binary task quality, a pull-and-filter task selection model (workers select which tasks they want to work on and the system filters these selections according to its optimization objective) and performs task allocation on the basis of worker accuracy measured in a [0,1] scale using a heuristic algorithm of linearithmic complexity. Although this study does incorporate the element of time, it is different than ours in that the model it uses assumes binary, homogeneous tasks rather than heterogeneous tasks of continuous quality. The use of homogeneous tasks (all tasks have the same difficulty, no distinction of task topics) means that optimization needs to be performed in terms of the number of jobs per worker, rather in terms of allocating specific workers to specific jobs according to their skills. Bernstein et al. [4, 3] also work with homogeneous microtasks and propose a retainer model for pre-recruiting (reserving) the optimal number of workers, so as to minimize task completion latency. This work however does not take into account worker skills and subsequently does not seek to maximize task quality. On heterogeneous microtasks, Faridani et al. [13] and Minder et al. [33] add the element of pricing to the problem model, proposing task pricing algorithms that aim to maximize the number of tasks finishing on time (the first study), while respecting budget and quality constraints (the second study). Mechanism design [35, 36] and multi-armed bandit mechanism design [5] mechanisms have also been employed to manipulate the time behavior of the crowd towards an efficient execution of time-critical tasks. The above works are different from our work, in that they do not explicitly sequence the tasks to the workers but they rather seek to incentivize the crowd's timely responses in order to achieve the time-related task objective.

In regards to *time-sensitive macrotasks* Khazankin et al. [26] propose a mathematical optimization approach that learns the task selection behavior of workers and then executes tasks in a manner that optimizes for cost and considers deadlines. This approach does not consider task quality, yet it is one of the first attempts to sequence time-sensitive macrotasks. Finally, Boutsis and Kalogeraki [7] propose an multi-objective optimization approach which searches for Pareto-optimal solutions, seeking to identify the group of workers (among multiple candidate groups) with the highest probability of finishing the task on time. This approach is different than ours, in that it does not apply sequencing along a timeline, but rather makes one-shot assignments based on the worker probabilities of meeting the deadline.

Overall, crowdsourcing optimization studies have so far examined mainly the assignment but not the sequencing aspect of the problem. The works that optimize for time-sensitive task characteristics are few and they either focus on binary/n-ary microtasks (which differ significantly from the expert macrotasks that we are interested in) or they do not sequence the worker-to-task assignments along a timeline. We address in our work the problem of optimal task sequencing for *online*, *heterogeneous*, *time-constrained*, *macrotasks*. As we will see in the following section, it is this type of tasks that expert crowsourcing consists of, and thus their optimization has significant impact on many recent platforms and applications.

3 Task Assignment and Sequencing (TAS)

In this section we first describe, from a high-level viewpoint, the expert crowdsourcing task model which we target through this work. Then we provide an example to illustrate the importance of adding the timeline sequencing element into the above setting. Next we define the TAS problem model, in terms of the input data, feasible solutions, constraints and optimization goal. Finally we analyze this problem model in terms of complexity.

3.1 Expert Crowdsourcing Setting

The expert crowdsourcing problem setting, at which this work is aimed, features some very particular characteristics that make it unique compared to other crowdsourcing problem settings:

1. **Heterogeneous rather than homogeneous tasks**. We work with crowdsourcing tasks that require different skills and skill levels from the workers, and that belong to multiple "topics" (rather than a single one). Workers in this setting possess a set of skills, and are less replaceable and less abundant than crowdsourcing settings that consider homogeneous tasks and skills (everyone can do every task).

2. **Macro- rather than microtasks.** Whereas microtasks accept binary or n-ary (multiple choice) worker input, and their quality is defined by assigning multiple simultaneous workers on the task and then performing majority voting, macrotasks feature open-ended worker input (e.g. write a product description), and their quality is built by one worker contribution after the other (sequentially rather than simultaneously).

3. **Online rather than offline.** Rather than working on a simplified offline setting, where the pool of workers and/or tasks are a-priori known, we consider an online setting, where workers demonstrate a dynamic flow of arrivals and departures and tasks arrive in an unpredictable manner. Any sequencing decision must be made based on task/worker information available up to the specific point in time.

4. **Time-constrained rather than only cost/quality-constrained tasks.** In addition to the need of achieving a certain utility metric (e.g. quality, number of acceptable tasks etc.) and the need to keep costs within budget, the online macrotasks of this setting have a deadline, i.e., they must also finish by a specific time point.

Application areas. Many tasks and platforms, especially recent ones, fit the above *expert crowdsourcing* setting and could benefit from its optimization. The first example are platforms such as upWork[4] that work with freelancer experts on creative tasks such as web design and development, document writing, or coding. Social innovation platforms such as OpenIDEO[5] or creative product design platforms like Quirky[6], where users build on one another's ideas, could also directly benefit from the optimization of the above setting. Finally collaborative document editing applications, such as corporate wikis [29], where it is possible to sequence worker contributions along a timeline could significantly improve from our approach.

3.2 The importance of the time element

Before giving the formal definition of the TAS optimization problem, we illustrate through an example the importance of adding the perspective of time, and how this addition has a significant on performance in expert crowdsourcing.

Example. Suppose there are only two jobs given, both from the same knowledge domain. Each job $j = (Q, C)$ has a quality threshold Q that needs to be reached, and a cost threshold C that must not be

[4]https://www.upwork.com/
[5]https://openideo.com/
[6]https://www.quirky.com/

exceeded. For this example suppose

$$j_0 = (5, 5) \text{ and } j_1 = (4, 4).$$

On the other hand, each worker $i = (e, w)$ has an expertise e that increases the quality of a job, and a wage w that consumes this job's budget. Let us assume that three workers

$$i_0 = (2, 3), i_1 = (3, 2) \text{ and } i_3 = (2, 1)$$

are given. Then each job has two possible assignments within budget and with sufficient quality:

$$\begin{array}{rcl} j_0 & : & \{i_0, i_1\} \text{ or } \{i_1, i_2\} \\ j_1 & : & \{i_0, i_2\} \text{ or } \{i_1, i_2\} \end{array}$$

For both jobs the latter assignment seems to be preferable over the other since workers $\{i_1, i_2\}$ provide more quality for less cost. Now we look at a sequencing period of three timeslots with limited worker availability as follows (both jobs are released immediately):

timeslot	0	1	2
i_0			×
i_1		×	
i_2	×		×

Since worker i_1 is available only at a single timeslot it is clear that at most one job can realize the preferable assignment mentioned above. So assume for the moment that we choose modestly $j_0 \leftarrow \{i_0, i_1\}$ for j_0. This gives us the following partial schedule for the workers:

timeslot	0	1	2
i_0			j_0
i_1		j_0	
i_2	×		×

But now none of the two feasible assignments for j_1 can be realized since only worker i_2 remains available. Although the assignment $j_1 \leftarrow \{i_2\}$ is within budget, it does not reach the needed quality, and j_1 remains incomplete in this case.

So let us choose the alternative assignment $j_0 \leftarrow \{i_1, i_2\}$ and set i_2 on j_0 at timeslot 0:

timeslot	0	1	2
i_0			×
i_1		j_0	
i_2	j_0		×

Now j_1 cannot be completed without violating the sequential working assumption w.r.t. this job. On the other hand, if we set i_2 on j_0 at timeslot 2, we can complete both jobs with the schedule

timeslot	0	1	2
i_0			j_1
i_1		j_0	
i_2	j_1		j_0

without violating any constraints. To complete the discussion, note that if we choose $j_1 \leftarrow \{i_1, i_2\}$ in the beginning, then j_0 cannot be completed no matter what timeslot is used for i_2 **(end of example)**.

The example shows that not only the choice of an optimal worker-task assignment without consideration of time may be misleading, but also that the specific selection of timeslots is important.

3.3 The TAS Problem Model

With the following definition we want to capture the interplay between task assignment *and* timeline sequencing within the same model, and add appropriate constraints. We refer to our problem description as TASK ASSIGNMENT AND SEQUENCING (TAS) in expert crowdsourcing.

3.3.1 Input Data

Scheduling period. Suppose we look at t timeslots $[t] = \{0, 1, \ldots, t-1\}$. Each timeslot $d \in [t]$ is also called a *day* but can be any fixed period of time.

Knowledge domains. A finite set K of knowledge domains. Each $k \in K$ represents an area of knowledge or a knowledge topic.

Workers. A finite set U of users, hereby refered to as workers, participating in the crowdsourcing platform. Each worker $i \in U$ has the following characteristics[7]:

[7]Note that in the context of this work the quantification of worker expertise, wage or speed are considered orthogonal

- *Expertise*. An expertise vector e_i of dimension $|K|$. The expertise e_{ik} of a worker denotes the added quality that the worker can bring to a job belonging to domain k.
- *Wage*. A cost vector w_i of dimension $|K|$. The amount w_{ik} is the monetary renumeration that the worker demands in order to perform a job belonging to domain k.
- *Availability*. An availability vector a_i of dimension t with entries $a_{id} = 1$ if worker i is available on day d, and $a_{id} = 0$ otherwise.

Jobs. A finite set J of knowledge-intensive jobs that are crowdsourced. A job j is assumed to have the following characteristics:

- *Domain*. Each job belongs to exactly one domain $k_j \in K$.
- *Quality threshold*. The amount Q_j is the minimum quality that the job needs to achieve.
- *Cost threshold*. The budget for job j is given by C_j as the maximum total amount of money that can be paid for the job.
- *Release date*. Each job has a release date $r_j \in [t]$ which means that job j enters the crowd system at timeslot r_j (and never leaves the system).

Sequentiality. Finally our model assumes a *sequential work mode* along the timeline, according to which workers build on one another's contributions, at most one worker can be assigned to a task simultaneously, and each worker contributes at most to a single task at a time. Sequentiality is chosen for three reasons. First it is often imposed by the nature of expert crowdsourcing macrotasks, which are not easily decomposable to microtask level and as such they do not allow multiple simultaneous worker contributions (e.g. writing a document cannot be done by decomposing it to sentence level). Second, sequentiality allows building on the task's quality while not necessitating worker concurrency, which in practice is more difficult to achieve when specific worker skills (i.e. experts on a topic) are required. Third, sequentiality allows a more realistic coupling of our approach with worker skill evaluation mechanisms, making it easier to accurately evaluate the quality that each worker has brought once she has finished working on a task. Nevertheless, as also discussed in section 6 an extension of our model to include worker concurrency is feasible and we aim to examine it as part of our future work.

to the studied assignment and sequencing problem. The interested reader is referred to [29, 8, 19] for available worker profile quantification techniques based on machine-learning, implicit evaluation or information theory.

3.3.2 Feasible Solutions, Constraints and Optimization Goal

A schedule needs to carry information about the resource allocation for each job in terms of workers and in terms of time: When does what worker contribute to which job?

Solutions. In a solution for input data $x = (t, K, U, J)$ we have for each job $j \in J$ a vector U_j of dimension t with entries from $U \cup \{none\}$. If $U_{jd} = i$ then worker i is assigned to job j and scheduled on day d, and if $U_{jd} = none$ then there is no worker assignment for job j on day d.

Note that we represent solutions hereby as job/timeslot-schedules with worker entries, whereas in the previous example we utilized an equivalent worker/timeslot-representation with job entries. So the successful schedule from the example in the present notation is

$$\begin{aligned} U_0 &= (none, i_1, i_2) \\ U_1 &= (i_2, none, i_0) \end{aligned}$$

Constraints. A solution is called *feasible* if and only if the following holds:

(a) No worker is assigned to more than one job at a time, i.e., for all $d \in [t]$ and distinct $j, j' \in J$ it holds that $U_{jd} \neq U_{j'd}$ (unless both values are none).

(b) No job is assigned to more than one worker at a time, i.e., for all $j \in J$ and $d \in [t]$ there is at most one worker stored in U_{jd}. This is ensured by the representation of U_j.

(c) No worker is assigned more than once to the same job, i.e., for all $j \in J$ and distinct $d, d' \in [t]$ it holds that $U_{jd} \neq U_{jd'}$ (unless both values are none).

(d) No worker is scheduled on a day where she is not available, i.e., for all $d \in [t]$ and $j \in J$ it holds that if $U_{jd} = i$ then $a_{id} = 1$.

(e) No job is worked on before its release date, i.e., for all $j \in J$ and $d < r_j$ it holds that $U_{jd} = none$.

(f) No job exceeds its budget, i.e., for all $j \in J$ it holds that $c_j \leq C_j$ where c_j is the *cost of job* j defined as $c_j = \sum_{i \in U_j} w_{ik}$ if j has domain k.

Note that there always exists a trivial feasible solution with $U_{jd} = none$ for all j, d.

Objective. In order to assess the quality of a feasible solution $y = \{j \mapsto U_j \mid j \in J\}$ we determine for each j with domain k the *quality of job* j w.r.t. this solution as $q_j = \sum_{i \in U_j} e_{ik}$. Note that in the context of this work we define task quality as the sum of expertises of the workers that participate in it, using the additive skill aggregation model that is often used for expert sequential macrotasks such as document editing [1, 31]. Other aggregation functions, including minimum, maximum or product [1] could also be used to compute a task's quality, however their full examination is out of the scope of this work.

Now we set the measure for input $x = (t, K, U, J)$ and solution y to

$$m(x, y) = |\{j \in J \mid q_j \geq Q_j\}|$$

which we want to maximize. Therefore we count the number of jobs that reach their quality threshold within budget and that can be scheduled in a feasible way w.r.t. constraints (a) to (f). We call such jobs *completed*.

3.4 TAS: An allocation problem with two aspects

The TAS optimization problem combines aspects of two well-studied problems of different nature, reflecting resource allocation of workers on one hand, and allocation of timeslots on the other.

3.4.1 Allocation of Workers: The Multiple Knapsack perspective

If we look only at worker allocation in our model, we can understand each job j of domain k with budget C_j as a knapsack of this size that we need to fill with worker's expertises e_{ik}. Since the worker availabilities restrict the number of times a single worker can be packed, we have a *bounded* version of the MULTIPLE KNAPSACK problem [25]. The difference to this classical problem is the optimization goal. While in TAS we want to maximize the number of completed jobs with respect to their individual quality thresholds Q_j, the goal in MULTIPLE KNAPSACK is to maximize the sum of all packed expertises, no matter how these spread over the different knapsacks.

3.4.2 Allocation of time slots: The Openshop perspective

On the other hand, let us suppose a worker-task-assignment is already fixed such that all jobs reach their quality thresholds, and we need to schedule the selected workers along the timeline with respect to job releases and worker availabilities. Then we can understand this partial problem as a machine-scheduling problem: Here workers play the role of machines and jobs need to be processes on these machines. Observe that the order of processing is immaterial in our model, that we demand sequentiality, and that the processing time of a job on a certain machine is either 0 or 1 per timeslot (depending on whether the respecting worker is assigned to this job or not). So this aspect of TAS is a UNITTIME OPENSHOP problem with limited machine-availability and job release-dates [27]. Note that the adoption of a model also implies non-preemption, i.e. a worker cannot be interrupted once he/she has started working on a task. The goal of maximizing the number of completed jobs translates to minimizing the number of late jobs if we set t as the gobal deadline. We also want to mention that the sequencing of an already fixed worker-task-assignment can be reduced to the BIPARTITE LIST EDGE-COLORING problem [11]. Here jobs and workers form a bipartite graph with the worker-task-assignments as it's edges, and timeslots

are represented by colors. Then we assign a list of colors to each edge (j, i) such that worker i is available on these timeslots and job j is already released. A proper coloring of all edges can be found if and only if the previously fixed worker-task-assignment can be sequenced on the timeline.

3.4.3 TAS Complexity

Both aspects of TAS that we have pointed out above are NP-hard on their own, so is TAS as we show below. For an upper complexity bound note that the length of TAS-solutions is polynomially bounded in the input length and that the constraints can be checked in polynomial time if a solution is given, so TAS is an NP-optimization problem. Moreover, we observe that TAS is a large number problem, since it has KNAPSACK as a subproblem (if there is only a single job and each worker is available on a different single day). So it is reasonable to consider strong NP-hardness.

Theorem 1. TAS *is a strongly NP-hard optimization problem.*

Proof. We show NP-hardness with a polynomial-time many-one reduction from the NP-complete problem 3-DIMENSIONAL MATCHING [24]. For finite, disjoint sets X, Y and Z we say that $M \subseteq X \times Y \times Z$ is a 3-dimensional matching if for all distinct triples $(x_1, y_1, z_1), (x_2, y_2, z_2) \in M$ it holds that $x_1 \neq x_2$, $y_1 \neq y_2$ and $z_1 \neq z_2$. It is known that 3-DIMENSIONAL MATCHING is NP-complete even in the special case when $|X| = |Y| = |Z| = u$ and M has to be a perfect matching with $|M| = u$.

3-DIMENSIONAL MATCHING (3-DM)

Input: Finite and disjoint sets X, Y, Z with $|X| = |Y| = |Z|$ and a subset $W \subseteq X \times Y \times Z$.

Question: Is there a perfect 3-dimensional matching $M \subseteq W$?

Suppose an instance of 3-DM is given with $X = \{x_i \mid i \in [u]\}$, $Y = \{y_i \mid i \in [u]\}$, $Z = \{z_i \mid i \in [u]\}$ for some $u \geq 1$ and $W \subseteq X \times Y \times Z$. The idea is to use constraint (a) (no worker is assigned to more that one job at a time) to achieve the needed matching condition. We take elements from X (Y, Z) as workers available on day 0 (1, 2, resp.) and use domains to fix the given triples from W. More precisely, we define a corresponding TAS-instance (t, K, U, J) as follows:

- The scheduling period has $t = 3$ timeslots.
- There are $|W|$ many different domains in K.
- Each triple $w \in W$ is encoded as a job j_w, and all jobs have pairwise different domains. For all jobs j_w we set quality and cost threshold to $Q_{j_w} = C_{j_w} = 3$ and release date to $r_{j_w} = 0$.
- Workers are defined as $U = X \cup Y \cup Z$. For $x \in X$, $y \in Y$ and $z \in Z$ we set the availability to $a_x = (1, 0, 0)$, $a_y = (0, 1, 0)$ and $a_z = (0, 0, 1)$, respectively. To fix expertise and wage, we consider each triple $w = (x, y, z) \in W$ and the corresponding job j_w. If j_w has domain k then we define $e_{xk} = e_{yk} = e_{zk} = 1$ and $w_{xk} = w_{yk} = w_{zk} = 1$. All entries in expertise and wage vectors that are not addressed hereby are set to 0.

First observe that a job j_w with $w = (x, y, z)$ reaches it's quality threshold if and only if we assign workers $\{x, y, z\}$ to this job, since exactly these workers contribute to the job's domain.

Now we argue that the given 3-DM instance has a perfect matching M if and only if the constructed TAS instance has a feasible solution with $|M| = u$ completed jobs. If $M \subseteq W$ is a 3-dimensional matching, then we consider the TAS-solution $U_{j_w} = (x, y, z)$ for all $w = (x, y, z) \in M$. Since M is a matching all distinct solution vectors differ in all components, so constraint (a) is satisfied. All other constraints are easy to check, just note that each worker is available only on a single day, that all jobs are immediately released and that no job can exceed the budget. All jobs in this solution are completed due to our previous observation.

Conversely, note that if there is a feasible TAS-solution with completed jobs j_w and $w = (x, y, z)$, then it must be that $U_{j_w} = (x, y, z)$. Since constraint (a) holds, the solution vectors for any two distinct jobs differ in all components. So $M = \{(x, y, z) \mid U_{j_w} = (x, y, z) \text{ and } j_w \text{ completed}\}$ is a 3-dimensional matching and $|M| = u$.

The reduction function maps only to TAS-instances where all integer values are polynomially bounded in the input length, so strong NP-hardness follows. □

This rules out the possibility of pseudo-polynomial algorithms and the existence of fully polynomial-time approximation schemes for TAS unless P equals NP. Furthermore note that the reduction emphasizes the aspect of timeline sequencing, since worker-task-assignments in the constructed TAS-instance are trivial (there is exactly one feasible worker-assignment possible to reach the quality threshold of each job).

4 An Online Algorithm for TAS

Due to the dynamic nature of crowdsourcing systems, it seems not realistic to consider TAS as an *offline* problem where algorithms are provided with the complete input at once. In fact, worker availabilities are hardly predictable and it is usually not known in advance which jobs will enter the system at what time. So the problem of task assignment and sequencing is inherently *online* in nature and sequencing decisions have to be taken without complete information about the input data. We say that an algorithm for TAS has the *online property*, if it processes the input in a serial way w.r.t. the timeline $d = 0, 1, \ldots$ and in each step d the algorithm has to take its assignment decisions while having access only to the time-dependent information of the input for timeslots $\leq d$. These are the worker availabilities and the jobs released up to day d. For more background on the general concept of online algorithms we refer to [6].

To design such an algorithm we start with the following observation: Suppose a feasible solution y for TAS is given. If we look at a single day d in this solution then the assignments of workers to jobs for that day form a bipartite matching between the (uncompleted) jobs with (remaining) quality needs and budget on one hand, and the set of available workers for that day on the other hand. Constraints (a) and (b) form exactly this bipartite matching condition.

So conversely, if we proceed day by day with our online algorithm, we can try to compute a matching between the active (= released but incomplete) jobs J' in the system on that day, and the available workers U' for that day. Note that due to this choice of J' and U' we also immediately satisfy constraints (d) and (e). It remains to consider constraints (c) (no worker assignment to the same job twice) and (f) (no job exceeds it's budget). Both can be taken care of when we construct the edges of possible assignments in the bipartite graph between J' and U': If the remaining budget for a job is smaller than the wage of a worker in this domain, then the edge is omitted. The same is true if the worker has already been assigned to this job in the past. Both conditions can be checked when looking at the partial solution for timeslots $< d$. Together, this online procedure results in a series of matchings M_d for $d = 0, 1, \ldots, t-1$ that form a feasible solution y for TAS.

More than that, we want to choose a sequence of matchings that yields a large number of completed jobs. Among all possible matchings for each day d, which is the right one? We propose a greedy approach and compute in each step a matching, such that the sum of *profits* we get from the respective assignments for that day is maximized. More precisely, we construct for each day a *weighted* bipartite graph where each possible assignment (edge) claims a certain profit. In our algorithm, the profit is just the amount of expertise per wage unit (efficiency). The problem MAX WEIGHTED BIPARTITE MATCHING can be solved to optimality by known algorithms in polynomial time, e.g. if we apply the *Hungarian Method* this step has a running time proportional to $O((|J'| + |U'|)^2|E|)$ [28]. So we obtain the following online Algorithm 1 for TAS with polynomial running time $O(t|J|^3|U|^3)$.

This algorithm can be viewed as an online schema that allows multiple extension, which we discuss in the last section after some experimental evaluation using the present basic version.

Algorithm 1 TAS-ONLINE

Input: A TAS-instance $x = (t, K, U, J)$
Output: A feasible solution y for x.

1: *Set* $U_{jd} = none$ *for all* $j \in J$ *and* $d \in [t]$.
2: **for** $d = 0, 1, \ldots, t-1$ **do** ▷ proceed day-by-day
3: $J' = uncompleted\ jobs\ with\ r_j \leq d$ ▷ active jobs
4: $U' = workers\ available\ on\ day\ d$ ▷ active workers
5: $E = \emptyset$ ▷ edge set in bip. graph
6: **for** $(j, i) \in J' \times U'$ **do**
7: **if** $i \in U_j$ **then pass** ▷ ensures (c)
8: **if** $e_{ik_j} == 0$ **then pass** ▷ i has no expertise in j's domain
9: **if** $w_{ik_j} > C_j - c_j$ **then pass** ▷ ensures (f)
10: $profit \leftarrow e_{ik_j}/w_{ik_j}$
11: $E \leftarrow (j, i, profit)$
12: $M_d \leftarrow MaxWeightedMatching(J', U', E)$
13: **for** $(j, i) \in M_d$ **do**
14: $U_{jd} = i$ ▷ worker-task-assignment
15: **return** $\{j \mapsto U_j \mid j \in J\}$

5 Experimental Evaluation

With the hardness result we have already seen that TAS has the KNAPSACK decision problem as a subproblem. It is known from literature that no competitive algorithm for the *online* version of KNAPSACK exists where items arrive one at a time [32]. An online algorithm is called competive if the ratio of it's performance and an optimal offline algorithm's performance can be bounded, a usual performance measure for online algorithms [6]. It follows that no competitive online algorithm for TAS exists as well. Therefore, in order to evaluate TAS-ONLINE experimentally, we formulate alternative algorithms to compare with.

5.1 Experimental Setup

5.1.1 Benchmarks

We evaluate the performance of TAS-ONLINE using four benchmarks, with each benchmark extending the previous with a new functionality. The first version of the algorithm (RANDOM) builds a feasible solution randomly and without any individual preferences of workers that usually appear in a fully self-organized system. We simply iterate over the available workers and pick a feasible job.

Algorithm 2 RANDOM

Input: A TAS-instance $x = (t, K, U, J)$
Output: A feasible solution y for x.

1: *Set* $U_{jd} = none$ *for all* $j \in J$ *and* $d \in [t]$.
2: **for** $d = 0, 1, \ldots, t-1$ **do** ▷ proceed day-by-day
3: $J' = uncompleted\ jobs\ with\ r_j \leq d$ ▷ active jobs
4: $U' = workers\ available\ on\ day\ d$ ▷ active workers
5: **while** $U' \neq \emptyset$ **do**
6: *pick worker* $i \in U'$ *randomly, remove it*
7: $J'_i = feasible\ jobs\ for\ worker\ i$
8: *pick job* $j \in J'_i$ *randomly*
9: $U_{jd} = i$ ▷ worker-task-assignment
10: **if** $q_j \geq Q_j$ **then** *remove* j *from* J' ▷ remove completed jobs
11: **return** $\{j \mapsto U_j \mid j \in J\}$

To obtain the feasible jobs for i in line 7 we proceed as in lines 7 to 9 in TAS-ONLINE and additionally check that j is still without worker assignment for that day.

For the next version of the algorithm (RANDOM EGOISTIC) we assume that workers try to realize a larger wage with priority, thus modeling a typical crowdsourcing environment, where workers are self-appointed to tasks, trying to maximize their individual profit [37]. To do so, we substitute line 8 in the previous algorithm with the lines stated in Algorithm 3.

In the next step (RANDOM EGOISTIC FILTER), we extend RANDOM EGOISTIC with a filter that restricts

Algorithm 3 RANDOM EGOISTIC

80: *let* $k_0, k_1, \ldots$ *be the domains sorted decr. by i's wage*
81: **for** $k = k_0, k_1, \ldots$ **do**
82: J'_i = *feasible jobs for worker* i *from domain* k
83: **if** $J'_i \neq \emptyset$ **then**
84: *pick job* $j \in J'_i$ *randomly*
85: **break**

the jobs that are offered to each worker based on expertise. This models the practice employed by many crowdsourcing platforms today, where workers can only access a task if they successfully pass a "screening" (realized through the use of performance levels, golden data, reputation, or other means across the different platforms) [10, 21], which allows to expect a substantial contribution to the job's quality by these workers. This "screening threshold" is expressed by an additional parameter $0 < \mathit{factor} < 1$ which determines the minimal expertise needed. Therefore we additionally substitute line 7 with the following lines.

Algorithm 4 RANDOM EGOISTIC FILTER

70: J'_i = *feasible jobs for worker* i
71: *remove all* j *from* J'_i *with* $e_{ik} < (Q_j \cdot \mathit{factor})$

As as last variation of Algorithm 2 we choose a job for some worker completely deterministically with a greedy rule: Job j is assigned to worker i if the marginal contribution in terms of quality is maximal among all feasible jobs for worker i. This amounts to replacing line 8 in Algorithm 2 by the following line (an leave line 7 unchanged).

Algorithm 5 ONLINE GREEDY

80: *pick job* $j \in J'_i$ *such that* $e_{ik_j} - q_j$ *is maximal*

Note that all algorithms so far have the online property for TAS. Finally, for reasons of comparison, we use an *offline* algorithm (Algorithm 6) that does not have to proceed day-by-day but has access to the complete input at once. So this clairvoyant algorithm knows in advance what workers will be available on what days of the scheduling period, and also what jobs will eventually enter the system. It proceeds job-by-job and treats each job as a knapsack that has to be packed with workers (items) that are available after the job's release date. To obtain such a packing, it calls an optimal algorithm for MAX KNAPSACK that returns a packing with minimal cost such that the quality threshold is reached. Then, for the workers from this packing (= worker-task assignment), a sequencing on the timeline w.r.t. their availability is fixed, before the next job is considered.

The algorithm has two more parameters that influence the way workers are selected for input to the knapsack algorithm for a job j. With *lookahead* we specify the interval of timeslots $[r_j, r_j + \mathit{lookahead}]$ from which the available workers are chosen in order to control the maximum flow time of each job. Secondly, we use *minavail* to ensure that each worker has at least *minavail*-many free timeslots remaining in the above interval in order to facilitate the allocation of timeslots afterwards.

Algorithm 6 TAS-OFFLINE

Input: A TAS-instance $x = (t, K, U, J)$
Output: A feasible solution y for x.

1: *Set* $U_{jd} = \mathit{none}$ *for all* $j \in J$ *and* $d \in [t]$.
2: **for** $j \in J$ **do** ▷ ordered by release dates
3: U'_j = *feasible workers for* j
4: **for** $i \in U'_j$ **do**
5: **if not** *minavail* in $[r_j, r_j + \mathit{lookahead}]$ **then**
6: *remove* i *from* U'_j
7: $W_j \leftarrow \mathit{DynProgKnapsack}(U'_j, Q_j, C_j)$
8: **for** $i \in W_j$ **do** ▷ sorted decr. by expertise
9: d = *earliest available timeslot* $\geq r_j$ *for* i
10: $U_{jd} = i$ ▷ worker-task-assignment
11: $a_{id} = 0$ ▷ set i unavailable on d
12: **return** $\{j \mapsto U_j \mid j \in J\}$

The offline algorithm does not guarantee optimal solutions for TAS for various reasons. However, it is designed to complete as many jobs as possible by the particular use of offline information and by incorporating optimal solutions to the knapsack subproblems. Note that due to the standard dynamic-

programming (DP) algorithm for MAX KNAPSACK this is only a pseudo-polynomial time algorithm (the DP-table has dimension $|U| \times C_j$ for each job) [25]. While we observe that this still yields tolerable runtimes for realistic input sizes, it is also possible to scale down the range of cost thresholds, or to use a fully polynomial-time approximation-scheme instead, if runtime becomes crucial.

5.1.2 Evaluation Metrics and Experiments Overview

To evaluate our approach we compare the algorithms principally in terms of the objective function value (i.e. the metric that the TAS model is meant to optimize), both as an absolute number and as a percentage of the upper bound of completable jobs. We also use four auxiliary metrics, meant to provide more information on the algorithm's behavior: the number of assigned workers, flow time, budget utilization and quality reached.

We conduct two types of experiments: i) synthetic (sections 5.2 and 5.3), where we experiment with a known simulated crowdsourcing instance and its variations and ii) real-world (section 5.4), where we examine our model on an actual crowdsourcing platform.

5.2 Synthetic Data Experiment

5.2.1 Simulation parametrization

We first experiment with synthetic data, which were generated using the experimental result distributions reported in [2], where AMT workers worked on the complex task of news writing. For simplicity, all modeling elements were generated in the [0.1] scale. Worker expertise received a random value from a normal distribution with mean equal to 0.5 and a variance 0.15, while worker wage received a random value from a normal distribution with mean equal to 0.5 and variance 0.2. For this set of experiments worker acceptance was set equal to 1. Job quality threshold was modeled using a beta distribution with $\alpha = 5$, $\beta = 1$, so that most jobs require a quality of at least 0.6 of 1 and higher with only a tail of jobs requiring less. Job cost threshold was then modeled as linearly related to job quality. Worker and job arrivals were modeled as Poisson processes with an average $\lambda = 200$ worker/day, and $\mu = 20$ jobs/day, respectively. Overall, we simulated a timeline of 30 days, during which 1000 workers (re)entered the system and 600 jobs were requested, belonging to 10 knowledge domains. So we have the following numbers in terms of our model:

t	$\|K\|$	$\|U\|$	$\|J\|$
30	10	1000	600

5.2.2 Upper bound calculation

First we compute an *upper bound* on the number of ultimately completable jobs using the optimal DP-algorithm for MAX KNAPSACK: Assume for each job that this job is the first for which we compute a worker-task assignment, i.e., all workers with at least one available timeslot $\geq r_j$ are possible knapsack items regardless of any other assignments. Now if the DP-algorithm does not find a packing within budget and above the quality threshold with this input data, then this job cannot be completed whatsoever. For the present instance, it turns out that at most 515 out of the 600 jobs can be completed.

5.2.3 Quality experiments

We now conduct the quality experiments. In terms of the objective function, we observe that TAS-ONLINE does not reach the number of completed jobs of our offline algorithm, but that it is significantly better that the other online algorithms (cf. Table 1).

Table 1: Objective function (completed jobs)

Algorithm	absolute	% of bound
RANDOM	2	0,39
RANDOM EGOISTIC	98	19,03
RANDOM EGO. FILTER	114	22,14
ONLINE GREEDY	82	15,92
TAS-ONLINE	**355**	**68,93**
TAS-OFFLINE	411	79,81

To get a more precise picture, we want to compare these algorithms not only w.r.t. this single measure, but also look at other characteristics. Next we ask how many workers are assigned to each job, and how long the flow times are, i.e., the number of timeslots between release date r_j and the latest assigned worker for j (cf. Table 2). In both cases we take the average values over all jobs in the system (not only the completed ones).

Table 2: Number of assigned workers and flow time

Algorithm	workers	flow time
RANDOM	4, 77	4, 77
RANDOM EGOISTIC	3, 46	3, 46
RANDOM EGO. FILTER	1, 64	7, 37
ONLINE GREEDY	3, 56	2, 91
TAS-ONLINE	**3, 31**	**3, 31**
TAS-OFFLINE	2, 41	8, 11

In cases where both values are the same, we only have compact assignments per job without any free slots in between. While TAS-ONLINE seeks this type of assignments we note that TAS-OFFLINE creates notable slack times, presumably a price to pay for larger number of completed jobs.

Now we state how much budget is used with these assignments, and how much quality is reached, both relativ to the given thresholds and on average over all jobs in the system (cf. Table 3).

Table 3: Budget usage and reached quality in %

Algorithm	budget	quality
RANDOM	88, 16	60, 62
RANDOM EGOISTIC	92, 05	90, 27
RANDOM EGO. FILTER	52, 6	55, 77
ONLINE GREEDY	91, 44	87, 12
TAS-ONLINE	**94, 35**	**97, 76**
TAS-OFFLINE	67, 73	70, 21

Due to its greedy nature TAS-ONLINE reaches very high quality values including for incompleted jobs and exploits the given budgets to a large extend.

Finally, we show how the main performance measures develop over the time, see Fig. 1 for completed jobs and Fig. 2 for reached quality. Interestingly, we observe that the higher values in Fig. 1 for TAS-OFFLINE appear towards the end of the scheduling period. A possible explanation is that the lookahead mechanism of this algorithms takes the end of the timeline into account.

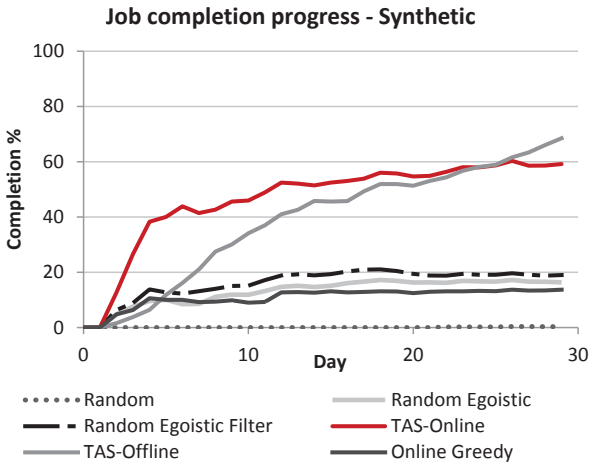

Figure 1: Job completion over time for each of the five tested algorithms. The proposed algorithm TAS-ONLINE manages to achieve more completed jobs most of the time compared to its competitors. Time unit expressed in days.

5.3 Scalability experiments

Next we perform a series of scalability experiments to examine the robustness of our proposed algorithm under varying conditions of the simulated instance. Given that worker volatility is the most uncontrollable factor in crowdsourcing, the two parameters that we vary are: the available expertise and the available number of workers. Each parameter is modified independently, while all the other parameters of the baseline instance presented in section 5.2 are kept the same. The variables that we measure are also the same as those measured for the baseline instance and

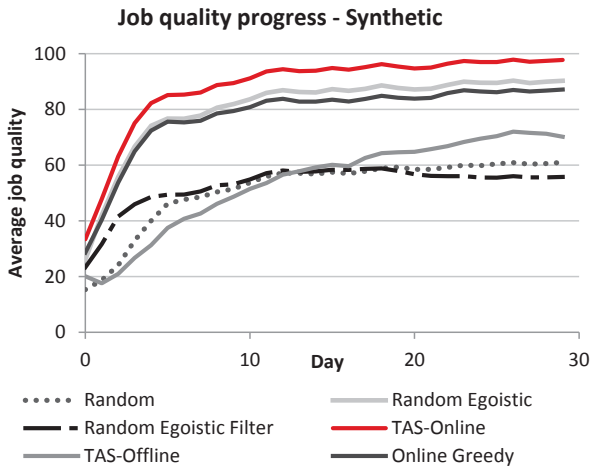

Figure 2: Average job quality over time for each of the tested algorithms. The proposed algorithm TAS-ONLINE manages to achieve higher quality per time unit compared to its competitors. Time unit expressed in days.

include the objective function, as well as budget utilization, total flow time, number of assigned workers per task and percentage of the average quality threshold reached.

The scalability experimental results are illustrated in Figures 3-5. For each of those figures the x axis corresponds to the varied parameter, the y axis to the measured variable and the vertical line at $x = 1$ corresponds to the results of the baseline instance reported in section 5.2.3.

5.3.1 Overview of scalability experimental results

Two main remarks can be drawn as an overview of the scalability experiments. The first is about *performance*: TAS-ONLINE is the highest performing among its online competitors, both regarding the value of the objective function, i.e. the metric that the algorithm is meant to optimize, and on quality, without significant compromises on any of the remaining metrics. TAS-ONLINE is the only algorithm among those examine to achieve this: whereas certain algorithms come close to its performance for certain metrics and parameter values, the same algorithms are significantly low-performing in other metrics and parametrizations. This result indicates that the proposed algorithm has a better 'value-for-money' compared to its competitors.

The second remark is about *consistency*: TAS-ONLINE is not only more performant, but its performance is consistent across the varying values of the scalability experimental parameters. This result indicates that the performance of the algorithm as detailed in section 5.2.3 is not incidental but an inherent property of the algorithm, and reinforces trust in the algorithm's future usage. In the following we present a detailed analysis of the scalability experiments.

5.3.2 Scalability effect on Objective Function: TAS-ONLINE gets consistently more jobs done

In Figure 3 we measure the value of the objective function (i.e. the number of accomplished jobs) as we modify expertise availability, and higher y axis values are better. As we can observe, the TAS family of algorithms (both the online and the offline version) are able to achieve and maintain higher performance than their competitive algorithms, at all expertise levels. For average expertise levels less than the baseline instance TAS-OFFLINE is the best-performing algorithm, followed closely by TAS-ONLINE, while for expertise levels slightly higher than the baseline TAS-ONLINE takes and maintains precedence. As it can be expected, as the average worker expertise per knowledge domain drops, the performance of all algorithms drops steeply as well. Nevertheless, we can also observe that the TAS algorithms are more robust, in the sense that they maintain their high performance when the other algorithms already start losing theirs (notice for example the almost unchanged performance of TAS-ONLINE between $x = 2$ down to $x = 1.2$ compared to the steep performance drop of the other algorithms in the same range).

A similar pattern can be observed when modifying the worker availability parameter (Figure 4). In this case too, TAS-ONLINE is by far the most performant of all the online algorithms, surpassed only by its offline

version. In fact, the performance difference between TAS-ONLINE and the rest of the algorithms is quite striking here, as TAS-ONLINE reaches approximately 60% of the objective function value while the rest of the algorithms only reach 20%. A second interesting remark that can be derived is that worker availability seems to have little effect on the algorithms after a certain critical mass of crowd workers has been gathered (which for our simulation corresponds to $x = 0.4$, i.e. 40% of the population of the baseline instance). These two observations (superiority of the TAS-ONLINE and small effect of worker availability after a certain critical mass) also hold when we measure the effect of the worker availability parameter on all other variables of the scalability experiment. Following this, and for reasons of brevity, we omit the rest of the scalability figures corresponding to the worker availability, and focus on the parameter of expertise availability which seems to have the highest effect.

5.3.3 Scalability effect on Quality: TAS-ONLINE achieves higher quality

Figure 5 illustrates the average task quality (expressed as the percentage of the quality threshold reached) for every level of expertise of the crowdsourcing population, and higher y axis values are better. We observe that TAS-ONLINE manages to achieve the highest quality levels, surpassing even its offline version, for all expertise levels. In fact, given a certain level of expertise ($x = 1.2$) and above, the algorithm manages to surpass the quality threshold set for the tasks. RANDOM-EGOISTIC and ONLINE GREEDY are the second and third most performing algorithms respectively, but unlike TAS-ONLINE they achieve their high quality results, at the cost of accomplishing too few jobs, as it can be seen by juxtaposing Figures 3 and 5.

5.3.4 Scalability on other parameters: TAS-ONLINE performs comparably to its competitors

Effect on cost. We now examine the effect that the modification of expertise availability has on the budget used by the allocation algorithms (Figure 6, smaller y axis values are better). As we may observe, TAS-ONLINE consumes most ($\approx 90\%$) of its available budget, at the same consumption level as the ONLINE GREEDY, RANDOM and RANDOM EGOISTIC algorithms. The RANDOM EGOISTIC FILTER and TAS-OFFLINE algorithms seem to make a slightly better usage of their budget. Nevertheless, the extra cost consumed by TAS-ONLINE is small, especially as expertise levels grow and more experts need to be paid (i.e. for $x > 1.2$). The significance of this extra cost gets even smaller considering what we gain in terms of the objective function (Figure 3), where TAS-ONLINE consistently accomplishes more jobs (almost up to double for $x = 1.2$) than RANDOM EGOISTIC FILTER. As such, TAS-ONLINE has a much higher 'value-for-mone' (jobs done vs. cost ratio) compared to its competitors.

Effect on Flow Time. Figure 7 shows the flow time of the algorithms for varying levels of expertise availability, and smaller y axis values are better. As we may observe, the proposed TAS-ONLINE algorithm behaves similarly to the rest of the online algorithms. This shows that there is no trade-off of performance for time, i.e. our algorithm does not achieve its higher objective function values at the cost of flow time.

Effect on Number of Assigned Workers. Figure 8 shows the change in the average number of workers per task, as we modify the availability of expertise, and lower values of the y axis are better. Here, and for most algorithms, we observe a very steep drop in the number of assigned workers, as the average expertise of the crowd worker population increases. This fact is to be expected, as the algorithms need to assign multiple workers to achieve the quality thresholds when expertise is scarce.

5.4 Real Data Experiment

To examine the effectiveness of the proposed algorithm, we conducted a real world experiment. The platform we used for this was CrowdFlower[8]. The task we used was collaborative news article writing, where workers from an initial hiring pool were asked to build on each other's content sequentially, enrich-

[8] http://www.crowdflower.com

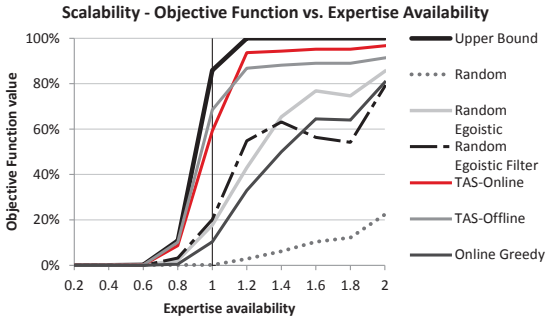

Figure 3: Objective function vs. expertise availability. The vertical line corresponds to the baseline simulated instance.

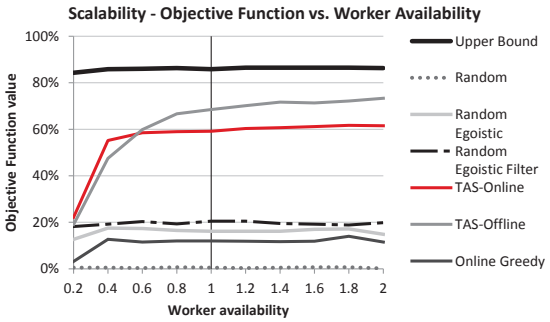

Figure 4: Objective function vs. worker availability.

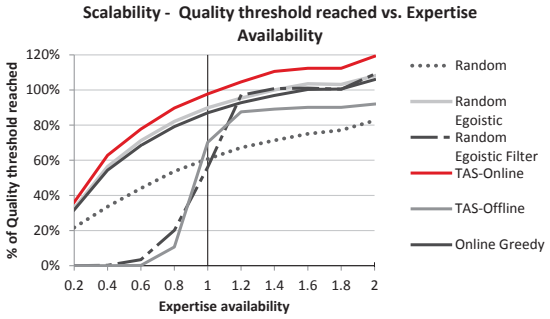

Figure 5: Quality reached vs. expertise availability.

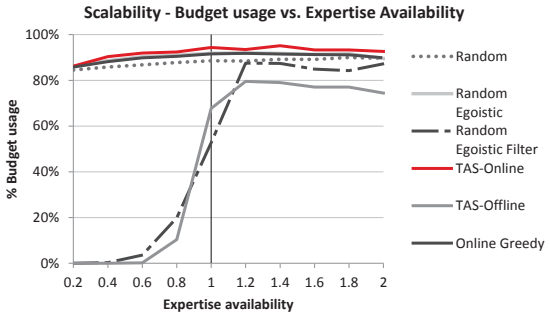

Figure 6: Budget availability vs. expertise availability.

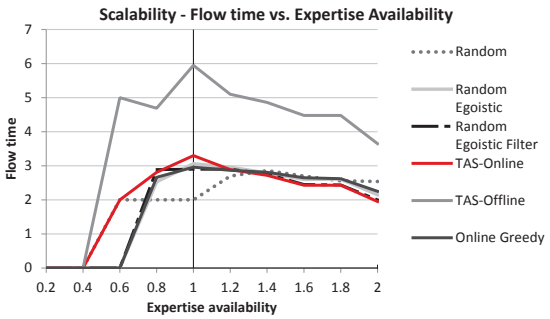

Figure 7: Flow time vs. expertise availability.

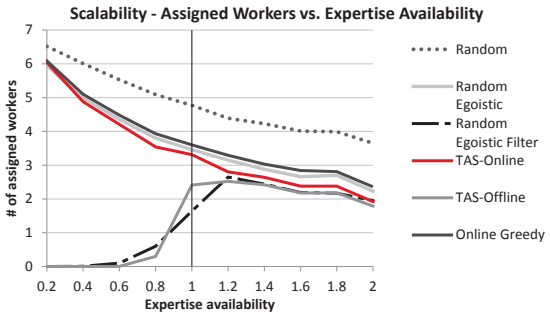

Figure 8: Number of assigned workers vs. expertise availability.

ing a news article text on a given topic. In more detail, our experimental workflow consisted of three steps.

In the first step we recruited a pool of 60 workers and recorded their expertise, wage and availability. To measure expertise, we asked each worker to complete two short multiple choice tests, each comprising 10 questions and measuring the workers' knowledge skills on a particular topic of current interest, i.e. "The FIFA 2015 corruption scandal" and "Self-driving cars" respectively. Each one of these topics is considered a knowledge domain for the purposes of our experiment. We also gave workers an estimation of the effort that they would have to spend on the second round of the task and asked them to provide us with their required wage.

In the next round we split the hired pool of workers randomly into two parts, one to be used by the benchmark and one by the optimization algorithm during scheduling. We also created six Google documents for each algorithm, three per knowledge domain, which corresponded to the jobs that would have to be accomplished. The quality and cost thresholds, as well as the release date for each job were set according to the same job generation criteria used in the synthetic experiments, and they were the same for the jobs of the benchmark and the optimization algorithm. In regards to the algorithms to be compared we used RANDOM EGO. FILTER as benchmark with $factor = 0.3$ (a worker was to be allowed to take a job only if his expertise was at least 30% of the target job quality) and TAS-ONLINE as the optimization algorithm. Finally we set the scheduling period to $t = 8$ slots and the time unit to one day. Each day, one worker would be invited to contribute to each Google document, according to the benchmark and the optimized algorithm. At the end of that day the document would be locked for the particular worker and sent for evaluation by a crowd of 50 independent crowd workers (different than those used in the experiments) to evaluate the job's current quality. Then, if the job had not surpassed its quality threshold and not exhausted its budget, a new worker was invited to work on the document.

At the end of the scheduling period, the results were as follows: The benchmark algorithm achieved a successful completion of 3 out of 6 jobs, while the optimization algorithm achieved successful completion of 5 out of 6 jobs. As it was expected the benchmark algorithm either allowed workers of the minimum necessary expertise to take a job, thus delaying the jobs quality progress too much, or it starved the job of budget. On the other hand TAS-ONLINE selected workers in such a way as to improve job completion within the given time period. As illustrated in Figures 9 and 10, similarly to the respective results of the synthetic experiments, TAS-ONLINE achieved higher average job quality and job completion percentages than the benchmark.

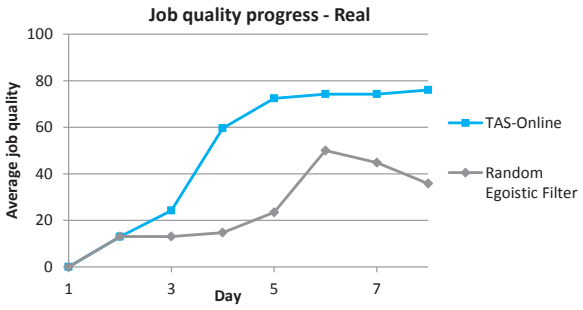

Figure 9: Average job quality over time. The proposed algorithm TAS-ONLINE manages to achieve higher quality per time unit compared to the benchmark. Time unit expressed in days.

6 Discussion and Future Extensions

The TAS model presented in this paper is the first concrete attempt to incorporate time-sensitive optimization in expert crowdsourcing. Our results, as presented in the previous, indicate that this model can improve the performance of crowdsourcing systems and help them utilize their human capital more effectively. Nonetheless, several challenges still lie ahead and many further extensions can be envisioned. In this final section we briefly discuss how the proposed TAS model and the TAS-ONLINE algorithm can be adapted to address further challenging settings

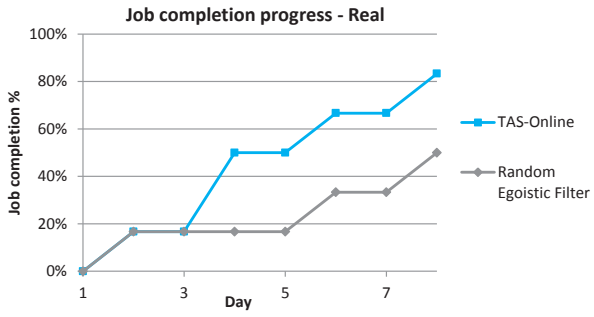

Figure 10: Job completion percentage over time. The proposed algorithm TAS-ONLINE manages to achieve higher job completion rates per time unit compared to the benchmark. Time unit expressed in days.

that may appear in practice.

Budget flexibility. Our initial TAS model assumes a fixed budget per job, set by the customers before the job is launched. However certain commercial platforms, like CrowdFlower allow jobs to go above their initial budget, and this option could be used in order to recruit better qualified workers during the scheduling period. One simple approach for adapting the TAS-ONLINE algorithm to the 'flexible budget option' is to recompute the daily matchings with alternating remaining budgets and explore their effects. A second adaptation is allowing edges to stay in the bipartite graph if the cost exceeds the remaining budget by a given fixed percentage, which can even be specified per job. Since we want to avoid that the algorithm makes too much use of the additional budgets we can reduce the profit of such edges accordingly. A multi-objective view of the optimization problem can also be useful to reveal these budget trade-offs.

Non-Acceptance of Assignment. The initial TAS model assumes that worker availability does not change, once the workers declare themselves available for a specific day. An adapted version of this model could be that workers can decline a certain assignment or they may be marked as unavailable by the system after a certain period, waiting for their reaction, has timed-out. A natural adaptation of the TAS-ONLINE algorithm to this problem is to perform a partial recomputation of the matching for the specific day, where all accepted assignments (workers and corresponding tasks) and all stalled workers are removed from the graph. Note that this also allows to bring in new workers and jobs that became available only very recently within the day.

Job prioritizing. The current TAS model gives all jobs the same priority. However some jobs may become more urgent than others during the scheduling period, for example in crowdsourcing systems dealing with crisis response [18]. Job prioritization can be incorporated in the model based on a given deadline, on the quality left to reach the threshold (jobs close to threshold go first), or other criteria. Our TAS-ONLINE algorithm uses a flexible profit function to rate all feasible assignments per day. Therefore if some jobs (or workers) become preferred over others, the profits on the respective edges can be easily changed. This change is possible for individual worker-task combinations and it can additionally be adjusted every day for a close progress control.

Quality aggregation model. Like many relevant studies in the area (e.g. [14, 2]), our current TAS model uses a sum function to calculate job quality, assuming that a job's quality is the sum of expertise of the individual workers that have contributed to the job. Nevertheless, and depending on the context, this calculation mode may not always reflect correctly a job's quality. For instance in cases of highly subjective tasks the quality of contributions of the same worker may vary even on tasks belonging to the same topic. In these cases, quality may need to be calculated in a different way, e.g. using crowd-based evaluations after each worker contribution, similarly to our real-world experiment. Here we need to distinguish two cases of adaptation for our algorithm. The first is incorporating a profit function that tries to forecast the benefit from a certain assignment, prior to that assignment. The second is incorporating an extra mechanism that is responsible for determining

the quality reached per job, day and assignment. The extra costs associated with this mechanism can also form part of the profit function. Both these adaptation are feasible depending on the exact aggregation model needed.

Learning. In our initial TAS model we consider expertise as an inherent, fixed property of each worker. For certain tasks however, like creative ones, the expertise of a certain individual can develop over time, and with the number of accomplished tasks, as workers 'learn by doing' [9, 38]. An improved version of the model could recognize this fact and perform an adjustment of worker expertise over time. According to this version, the expertise per worker needed by the TAS-ONLINE during graph construction each day could be the outcome of a previous learning process (e.g. machine learning as in [29]). The online version of the TAS algorithm is particularly well suited for such an dynamic adjustment.

Order of workers per job. In our problem formulation the order in which the assigned workers per job are placed on the timeline is arbitrary (openshop model), to simulate the first-come first-served mode of functionality of typical crowdsourcing platforms. It could nonetheless be reasonable for certain applications to require a specific order of workers, e.g. in a decreasing order of expertise. A straightforward approach to adapt TAS-ONLINE to such a request is to drop all the edges in the bipartite graph for each day such that the only workers that remain assignable are those that correspond to the order criteria. Again, this can be adopted over time such that, e.g., each job begins with an assignment of some workers with sufficient but relatively small expertise, to leave room and budget for enhancement.

Multiple assignments per worker and timeslot. Constraint (a) of our examined TAS model allows only one task per worker and timeslot. It may be reasonable to relax this to some bounded number of tasks per worker and timeslot, which can also be changed over time, for instance to allow multiple assignments to more experienced workers. Currently TAS-ONLINE relies on computing daily matchings, and the matching condition on the worker-side of the bipartite graph corresponds directly to constraint (a). However by taking a closer look at how the weighted matching are computed in terms of min-cost flow networks, we can observe that the capacities on source-edges and sink-edges in the flow network are usually set to 1 to enforce the matching conditions. If we now allow larger values as capacities on sink edges we can also compute worker-task assignments for the relaxed condition of multiple assignments.

The above correspond to the main modifications that could be made to adapt the proposed TAS model and TAS-ONLINE algorithm to multiple real-life situations, depending on the crowdsourcing platform, population and type of jobs at-hand. As such they could be used independently or in various combinations, as the starting points for further studies in this promising new field of expert crowdsourcing optimization.

7 Conclusion

In this paper we present TAS, a model that adds the timeline and online perspective to task assignment optimization in expert crowdsourcing. Using a greedy scheduling algorithm, TAS-ONLINE, we show that optimization under this model can significantly improve expert crowdsourcing performance. Future extensions to enable the TAS model to handle further challenging settings include: include adding budget flexibility, job and/or worker prioritizing, fine-tuning the job quality aggregation mechanism, worker performance variability over time, and multiple assignments per worker/timeslot.

8 Acknowledgments

The work of Ioanna Lykourentzou in this paper has received funding support by the Luxembourg National Research Fund (FNR) under INTER Mobility grant #8734708. The present paper is an extension of the work first presented at the Third AAAI Conference on Human Computation and Crowdsourcing

(HCOMP 2015) [30].

References

[1] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. Online team formation in social networks. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 839–848, New York, NY, USA, 2012. ACM.

[2] Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Task assignment optimization in knowledge-intensive crowdsourcing. *The VLDB Journal*, pages 1–25, 2015.

[3] Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David 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*, UIST '11, pages 33–42, New York, NY, USA, 2011. ACM.

[4] Michael S. Bernstein, David R. Karger, Robert C. Miller, and Joel Brandt. Analytic methods for optimizing realtime crowdsourcing. *CoRR*, abs/1204.2995, 2012.

[5] Arpita Biswas, Shweta Jain, Debmalya Mandal, and Y. Narahari. A truthful budget feasible multi-armed bandit mechanism for crowdsourcing time critical tasks. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 1101–1109, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[6] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 2005.

[7] I. Boutsis and V. Kalogeraki. On task assignment for real-time reliable crowdsourcing. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, pages 1–10, June 2014.

[8] Daniel Hasan Dalip, Marcos André Gonçalves, Marco Cristo, and Pável Calado. Automatic assessment of document quality in web collaborative digital libraries. *J. Data and Information Quality*, 2(3):14:1–14:30, December 2011.

[9] Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. Shepherding the crowd yields better work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1013–1022, New York, NY, USA, 2012. ACM.

[10] Julie S. Downs, Mandy B. Holbrook, Steve Sheng, and Lorrie Faith Cranor. Are your participants gaming the system?: Screening mechanical turk workers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2399–2402, New York, NY, USA, 2010. ACM.

[11] S Even, A Itai, and A Shamir. On the complexity of time table and multi-commodity flow problems. In *Foundations of Computer Science, 1975., 16th Annual Symposium on*, pages 184–193, October 1975.

[12] J. Fan, M. Zhang, S. Kok, M. Lu, and B. Ooi. Crowdop: Query optimization for declarative crowdsourcing systems. *Knowledge and Data Engineering, IEEE Transactions on*, PP(99):1–1, 2015.

[13] Siamak Faradani, Bjoern Hartmann, and Panagiotis G. Ipeirotis. What's the right price? pricing tasks for finishing on time. In *Human Computation*, volume WS-11-11 of *AAAI Workshops*. AAAI, 2011.

[14] Gagan Goel, Afshin Nikzad, and Adish Singla. Allocating tasks to workers with matching constraints: Truthful mechanisms for crowdsourcing markets. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 279–280, Republic and Canton of

Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.

[15] Daniel Haas, Jason Ansel, Lydia Gu, and Adam Marcus. Argonaut: Macrotask crowdsourcing for complex data processing. *Proc. VLDB Endow.*, 8(12):1642–1653, August 2015.

[16] Chien-Ju Ho and Jennifer Wortman Vaughan. Online task assignment in crowdsourcing markets. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[17] John Joseph Horton and Lydia B. Chilton. The labor economics of paid crowdsourcing. In *EC '10*, 2010.

[18] Muhammad Imran, Ioanna Lykourentzou, and Carlos Castillo. Engineering crowdsourced stream processing systems. *CoRR*, abs/1310.5463, 2013.

[19] Panagiotis G. Ipeirotis and Evgeniy Gabrilovich. Quizz: Targeted crowdsourcing with a billion (potential) users. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 143–154, New York, NY, USA, 2014. ACM.

[20] Hsu J. Jabbari S., Assadi S. Online assignment of heterogeneous tasks in crowdsourcing markets. In *Third AAAI Conference on Human Computation and Crowdsourcing (HCOMP-2015)*, 2015.

[21] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, March 2007.

[22] Saraschandra Karanam, Deepthi Chander, L. Elisa Celis, Koustuv Dasgupta, and Vaibhav Rajan. Adaptive performance optimization over crowd labor channels. In *Proceedings of the Seconf AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2014, November 2-4, 2014, Pittsburgh, Pennsylvania, USA*, 2014.

[23] David R. Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *CoRR*, abs/1110.3564, 2011.

[24] R M Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, January 1972.

[25] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer Science & Business Media, March 2013.

[26] R. Khazankin, B. Satzger, and S. Dustdar. Optimized execution of business processes on crowdsourcing platforms. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*, pages 443–451, Oct 2012.

[27] Svetlana A Kravchenko. On the complexity of minimizing the number of late jobs in unit time open shop. *Discrete Applied Mathematics*, 100(1-2), March 2000.

[28] H W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955.

[29] Ioanna Lykourentzou, Katerina Papadaki, Dimitrios J. Vergados, Despina Polemi, and Vassili Loumos. Corpwiki: A self-regulating wiki to promote corporate collective intelligence through expert peer matching. *Information Sciences*, 180(1):18 – 38, 2010. Special Issue on Collective Intelligence.

[30] Ioanna Lykourentzou and Heinz Schmitz. An online approach to task assignment and sequencing in expert crowdsourcing. Third AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2015), Works-in-Progress, nov 2015.

[31] Ioanna Lykourentzou, Dimitrios J. Vergados, and Yannick Naudet. Improving wiki article quality through crowd coordination: A resource allocation approach. *Int. J. Semant. Web Inf. Syst.*, 9(3):105–125, July 2013.

[32] A Marchetti-Spaccamela and C Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68(1-3):73–104, January 1995.

[33] P. Minder, S. Seuken, A. Bernstein, and M. Zollinger. Crowdmanager-combinatorial allocation and pricing of crowdsourcing tasks with time constraints. In *2nd Workshop on Social Computing and User Generated Content*. ACM Conference on Electronic Commerce (ACM-EC 2012), jun 2015.

[34] Luyi Mo, Reynold Cheng, Ben Kao, Xuan S. Yang, Chenghui Ren, Siyu Lei, David W. Cheung, and Eric Loz. Optimizing plurality for human intelligence tasks. In *CIKM13*, CIKM13, New York, NY, USA, 2013. ACM.

[35] Swaprava Nath, Pankaj Dayama, Dinesh Garg, Yadati Narahari, and James Zou. Mechanism design for time critical and cost critical task execution via crowdsourcing. In *Proceedings of the 8th International Conference on Internet and Network Economics*, WINE'12, pages 212–226, Berlin, Heidelberg, 2012. Springer-Verlag.

[36] V. Rajan, S. Bhattacharya, L. Celis, D. Chander, K. Dasgupta, and S. Karanam. Crowdcontrol: An online learning approach for optimal task scheduling in a dynamic crowd platform. In *ICML'13 Workshop: Machine Learning Meets Crowdsourcing*. ACM Conference on Electronic Commerce (ACM-EC 2012), jun 2015.

[37] Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*, 2011.

[38] S.B. Roy, I. Lykourentzou, S. Thirumuruganathan, S. Amer-Yahia, and G. Das. Crowds, not drones: modeling human factors in interactive crowdsourcing. In Reynold Cheng, Anish Das Sarma, Silviu Maniu, and Pierre Senellart, editors, *DBCrowd 2013 - VLDB Workshop on Databases and Crowdsourcing*, volume CEUR Workshop Proceedings, pages 39–42. CEUR-WS, 2013.

[39] Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Optimization in knowledge-intensive crowdsourcing. *CoRR*, abs/1401.1302, 2014.

[40] Han Yu, Zhiqi Shen, and C. Leung. Bringing reputation-awareness into crowdsourcing. In *Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on*, pages 1–5, Dec 2013.

[41] Tao Yue, Shaukat Ali, and Shuai Wang. An evolutionary and automated virtual team making approach for crowdsourcing platforms. In Wei Li, Michael N. Huhns, Wei-Tek Tsai, and Wenjun Wu, editors, *Crowdsourcing*, Progress in IS, pages 113–130. Springer Berlin Heidelberg, 2015.

[42] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. Task matching in crowdsourcing. In *Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, ITHINGSCPSCOM '11, pages 409–412, Washington, DC, USA, 2011. IEEE Computer Society.

[43] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. Task recommendation in crowdsourcing systems. In *Proceedings of the First International Workshop on Crowdsourcing and Data Mining*, CrowdKDD '12, pages 22–26, New York, NY, USA, 2012. ACM.

[44] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. Taskrec: probabilistic matrix factorization in task recommendation in crowdsourcing systems. In *Proceedings of the 19th international conference on Neural Information Processing - Volume Part II*, ICONIP'12, pages 516–525, Berlin, Heidelberg, 2012. Springer-Verlag.