Online Sequencing of Non-Decomposable Macrotasks in Expert Crowdsourcing

HEINZ SCHMITZ, Hochschule Trier
IOANNA LYKOURENTZOU, Luxembourg Institute of Science and Technology

We introduce the problem of Task Assignment and Sequencing, which models online optimization in expert crowdsourcing settings that involve non-decomposable macrotasks. Non-decomposition is a property of certain types of complex problems, like the formulation of an R&D approach or the definition of a research methodology, which cannot be handled through the "divide-and-conquer" approach typically used in microtask crowdsourcing. In contrast to splitting the macrotask to multiple microtasks and allocating them to several workers in parallel, our model supports the sequential improvement of the macrotask one worker at a time, across distinct time slots of a given timeline, until a sufficient quality level is achieved. Our model assumes an online environment where expert workers are available only at specific time slots and worker/task arrivals are not known *a priori*. With respect to this setting, 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 any future input information regarding job release dates or worker availabilities. Experimental results comparing TAS-ONLINE to five benchmarks show that it achieves more completed jobs, lower flow times, and higher job quality. This work bears practical implications for providing performance and quality guarantees to expert crowdsourcing platforms that wish to integrate non-decomposable macrotasks into their offered services.

CCS Concepts: • **Human-centered computing** → *Collaborative and social computing*; *Computer supported cooperative work*; • **Computing methodologies** → *Planning and scheduling*;

Additional Key Words and Phrases: Crowdsourcing optimization, cooperative social computing, online scheduling decisions, macrotask scheduling

ACM Reference format:
Heinz Schmitz and Ioanna Lykourentzou. 2018. Online Sequencing of Non-Decomposable Macrotasks in Expert Crowdsourcing. *ACM Trans. Soc. Comput.* 1, 1, Article 1 (January 2018), 33 pages.
https://doi.org/10.1145/3140459

1 INTRODUCTION

As the appeal of crowd work increases, there is a growing need to provide support for more complex tasks, such as document drafting, product design, social innovation, and idea development. Indicative of the interest that such tasks attract is their increasing inclusion to traditional crowd

The work of Ioanna Lykourentzou has received funding support from the Luxembourg National Research Fund (FNR), grant #8734708.
Authors' addresses: H. Schmitz, Hochschule Trier, Fachbereich Informatik, Schneidershof, D-54293 Trier, Germany; email: H.Schmitz@hochschule-trier.de; I. Lykourentzou, Luxembourg Institute of Science and Technology, L-4362 Esch-sur-Alzette, Luxembourg; email: Ioanna.Lykourentzou@list.lu.
The present article is an extension of the work first presented at the Third AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2015) (Lykourentzou and Schmitz 2015).

https://doi.org/10.1145/3140459

work platforms, such as CrowdFlower (see the recently launched CrowdFlower Labs[1]), but also the proliferation of dedicated platforms, such as upWork[2] and crowdSpring.[3] This type of crowdsourcing is often referred to as expert crowdsourcing (Retelny et al. 2014), and the tasks that it involves are referred to as macrotasks (Cheng et al. 2015; Haas et al. 2015). Macrotasks differ from the typically crowdsourced microtasks in that they often require expert skills, assume varying degrees of knowledge over a topic, may take more worker time, and often involve worker collaboration, i.e., workers building on each other's intellectual contributions.

The current trend in crowd-powered applications research is to process macrotasks using a "divide-and-conquer" approach: Macrotasks are decomposed to smaller (microtask-level) units, parallelized (to multiple workers) and then recomposed (to form the final answer from the separate smaller subtasks). Examples include captioning partial snippets of speech and then merging them into a single output stream (Lasecki et al. 2012), categorizing individual items into a small number of categories and then merging these categorizations to create taxonomies (Chilton et al. 2013), or aggregating multiple word or sentence-level translations to form a larger corpus (Ambati et al. 2012; Zaidan and Callison-Burch 2011). Although often very effective (Teevan et al. 2016), this is a costly process as it requires the intervention of experts to define—through tailor-made workflows—how the macrotask should be broken into smaller chunks of work, which dependencies exist among these chunks, and how these can be recomposed once complete (Kim et al. 2014; Chan et al. 2016). The necessity for expert involvement in the workflow design has been proven even in cases where the crowd was considered, in an effort to lower costs (Kittur et al. 2011; Kulkarni et al. 2012). Further, breaking a macrotask to smaller work units is feasible for some but not all types of macrotasks: Borrowing from organizational literature, many R&D problems are non-decomposable (Felin and Zenger 2014), meaning that they cannot be broken down to smaller sub-problems, which can be solved in parallel. For such tasks, no crowdsourcing optimization model exists to date; current literature suggests that they can only be solved in-house and manually instead of posting them even to specialized crowd platforms like Innocentive (Sieg et al. 2010). Finally, since different macrotasks have different needs and serve different purposes, making a one-size-fits-all workflow for task decomposition, parallelization, and decomposition is difficult, inevitably rendering the macro-to-micro task logic less generalizable.

In this work, we pose the following question: What if decomposition is not the only way to handle complex crowd work? With this in mind, we propose a new crowd work model, one that assumes *non-decomposable macrotasks* that get *iteratively improved one worker at a time* until a sufficient quality level is achieved. Our model is not based on worker parallelization but on worker sequencing across time slots. A given worker takes a lock on a given macrotask for the duration of time (slot) that this is assigned to him/her. When time is up, the task is assessed against a quality threshold. In case this threshold is not reached, the next worker is selected, takes a lock, and works on the task. Parallelization in our model happens at the level of the task batch, with multiple workers owning and working on different macrotasks during the same time slot. Optimization shifts from single-slot assignment decisions ("which workers should work on which task?") to multiple-slot assignment and sequencing ones ("which worker should work on which task and on which time slot?"). By sequencing assignment decisions and not assuming specific task splitting and dependencies, our model becomes more easily generalizable.

This shift distinguishes our model from previous works. Recent studies in expert crowdsourcing optimization (Basu Roy et al. 2015; Goel et al. 2014) typically seek to find multiple worker

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

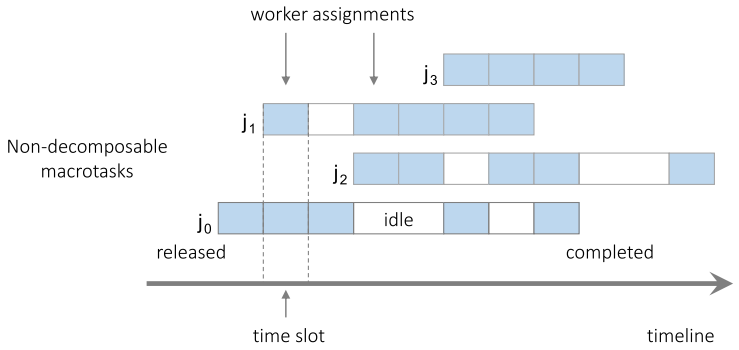

Fig. 1. TAS problem model overview. TAS is based on online, non-decomposable, heterogeneous macrotasks, each of which is iteratively improved one worker at a time. TAS combines two optimization sub-problems: (i) assignment (which worker to assign to which macrotask per time slot) and (ii) sequencing (how to dynamically roll these per-slot assignments across a realistic timeline).

assignments per task such that the worker contributions add up to a required quality threshold within a given budget. The assignments all take place in the same, single time slot. This model comes with two shortcomings. First, it is not applicable in cases where the task (or the part of the task that we are interested in) is non-decomposable. Second, it does not inherently handle time-dependent elements such as the availability constraints of the expert workers at different time periods, or the time-differentiated task arrivals, unless a lookahead mechanism has been put in place. In fact, the assumption of a unique time slot often leads to the adoption of an offline setting, according to which the optimization algorithm can access the complete expected task and worker input and use it to compute an index of pre-calculated assignments. In a realistic crowdsourcing setting, however, worker/task arrivals and departures may vary across the timeline in an unpredictable manner. Handling non-decomposable tasks across an unpredictable timeline thus raises the need to solve not only the *assignment part* of the worker allocation problem (finding which worker should take which task) but also the *sequencing part* (identifying when each worker should contribute). It also raises the need for online algorithms, which can make efficient decisions having access only to information that is available until their decision point and revisit these decisions frequently to adapt to changes in the worker/task availability, rather than algorithms that use pre-calculated assignments.

In this article, we introduce the problem of crowdsourcing Task Assignment and Sequencing (TAS), which shifts the crowdsourcing optimization model from single-slot parallel assignments to multiple-slot sequencing ones (Figure 1). The main question that TAS evokes is as follows: Which worker-to-task assignments are the best (assignment problem), and how can we optimally roll them out in a realistic timeline (sequencing problem), featuring unknown task release dates and worker availabilities?

To the best of our knowledge, this is the first work that addresses the problem of assignment and sequencing optimization for expert non-decomposable crowdsourcing macrotasks. Overall, our three main contributions with this article are as follows:

—We propose a new crowd work optimization model, which makes a conceptual shift from task decomposition and worker parallelization to task non-decomposition and worker sequencing. In contrast to current models that assume multiple workers per task on a single time slot, our model is based on the idea of a single worker-per-task-per-time-slot rolled across multiple slots. Our model combines assignment decisions (per time slot and across

tasks) with sequencing decisions (rolling out these assignments along a timeline) under reasonable constraints. We call this problem model TAS and prove its strong NP hardness.
— We propose an online algorithm, TAS-ONLINE, which operates on this new model and seeks to complete as many jobs as possible within budget, required quality and given timeline. At each time slot, the algorithm computes a feasible matching of workers to tasks for the next slot, without any future input information regarding job release dates or worker availabilities. This results in feasible sequences of workers per task up to the present time slot.
— We illustrate, through simulated and real-world experiments, that TAS-ONLINE can achieve more completed jobs, lower flow times and higher quality compared to five typical benchmarks.

The rest of this article is organized as follows. In Section 2, we recapitulate the related literature on crowdsourcing optimization, distinguishing between works that focus on decomposable tasks and works that indicate the need for non-decomposable ones. In Section 3, we describe the characteristics of the expert crowdsourcing setting that this work applies to. We also illustrate, through a mathematically formulated toy example, why sequencing across time slots 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 five 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 to 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 article'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 (Karanam et al. 2014), discovering the optimal worker wage (Horton and Chilton 2010), 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 that is applicable on n-ary tasks with an objective true value) (Mo et al. 2013), optimizing the worker queue size (Bernstein et al. 2012, 2011), determining the optimal task pricing to incentivize the crowd's timely responses (Faradani et al. 2011; Minder et al. 2012; Nath et al. 2012; Rajan et al. 2013; Biswas et al. 2015), or identifying the optimal set of tasks to forward to the crowd (for systems like database query execution that are based partially on crowdsourcing and partially on automated methods) (Fan et al. 2015).

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, 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, *assignment* and *sequencing*. The assignment problem translates to solving which worker should be given which task. The sequencing problem adds the existence of a timeline and translates to solving when the worker will be given the task.

Current studies, which mostly assume a decomposable task model, focus on the assignment part, i.e., on selecting the optimal set of workers that will work in parallel on the task. When

timeline is taken into account, this serves to predict future worker behavior and thus improve the selection of the best assignment, rather than to define the specific time slots that the workers will undertake the task. Below, we first review the literature on decomposable task optimization and then go through the studies that stress out the need to also account for non-decomposable tasks.

2.2 Decomposable Tasks

2.2.1 Optimizing Task Assignments. Karger et al. (2011) are among the first to examine assignment optimization in crowdsourcing. Working with homogeneous microtasks (that all have the same level of difficulty and do no distinguish among task topics), they 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 also 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 possible assignments of workers to tasks. Ho et al. (2012) work with heterogeneous microtasks of n-ary classification quality on a model where worker skills per microtask are unknown. They 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. (2012a, 2012b, 2011) 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. Finally, Bragg et al. (2014) highlight the need for sustaining worker engagement and propose a parallel task routing algorithm that makes assignments to all available workers in the platform for a near-optimal use of the available crowd workforce. Working with macrotasks divisible to independent subtask level, Goel et al. (2014) and Roy et al. (2015, 2014) 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. (2015) 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.

The difference of these works with ours, which also renders the optimization algorithms that they propose not directly applicable on our setting, is that they model one-time parallel and not multiple-slot sequential assignments.

2.2.2 Considering the Timeline. Further to pure cost/quality assignments, certain works consider the existence of a timeline with varying worker/task arrivals and departures. When included in the model, the timeline serves to learn the time-dependent behavior of the workers and select the task assignments that increase the probability of the task finishing on time. Sequencing workers across the timeline is not optimized for, since the decomposable task model still requires only one-time assignments. In this context, Boutsis and Kalogeraki (2014) propose a multi-objective optimization approach that searches for Pareto-optimal solutions, seeking to identify the group of workers (among multiple candidate groups) with the highest probability of finishing the task promptly. Khazankin et al. (2012) propose a mathematical optimization approach that learns the task selection behavior of workers and then executes tasks in a manner that optimizes for cost (but not quality) and considers deadlines. Among the few works that split the timeline into distinct time slots is the study by Yu et al. (2013). This study optimizes the number of tasks to recommend to each worker per time unit with the objective of maximizing average number of successful (i.e., of acceptable quality) jobs for a given time period. Their model assumes binary microtasks and a pull-and-filter task selection model (where 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 the model of this study does include time slots, it is different than ours in that it assumes homogeneous tasks of binary quality 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.

The difference of the above works with ours is on the role of the timeline. When the task is decomposable and multiple workers can be assigned on it at the same time slot, like in the above works, time serves to select the best assignment, i.e., the set of workers that (on top of fulfilling cost and quality requirements) can finish the task as fast as possible or according to future time demands, based on a lookahead mechanism. When the task is non-decomposable, like in our model, the timeline can be used to divide time into distinct slots (each with a fixed and limited horizon) and select the best sequencing of workers who can fulfill the cost and quality requirements across these slots and before the timeline ends.

2.3 Non-Decomposable Tasks

2.3.1 Current Usage in Crowdsourcing. Apart from decomposable tasks, which form the majority of current crowdsourcing optimization models, very recent works have started to indicate the need for non-decomposable tasks. In this context, Kim et al. (2017) argue that certain types of complex creative tasks, like, for example, story writing, cannot be easily split into independent tasks because of the many interdependencies and the need to maintain the global context. Consequently, they propose a sequential workflow that is based on iterative reflection and high-level revisions of the work and show that it can be an effective complement to existing parallel crowdsourcing techniques. In the same spirit, Parameswaran et al. (2016) indicate that open-ended crowdsourcing problems, where there is a need to incorporate multiple viewpoints and find the common ground, are the "next frontier" in crowdsourcing research. They also indicate that such problems need to be treated as single open-ended crowdsourcing tasks, since decomposing them to "Boolean crowdsourcing" microtasks would mean losing information related to the problem's broader context. Finally, Little et al. (2010) also provide early support that certain open-ended tasks, like writing, profit the most from iterative sequential rather than parallel workflows. Despite acknowledging the need for sequentiality in handling non-decomposable crowdsourcing tasks, no work to date examines the optimization of such a model.

2.3.2 Positioning Non-Decomposable Tasks within a Broader Task Model Taxonomy. To understand better the nature of non-decomposable tasks, and the reasons that mandate their handling according to a sequential assignment model, it is important to first position them with respect to other crowdsourcing task types and the problems that each type can solve. Drawing from organizational management literature and adapted to the crowdsourcing landscape, Table 1 classifies crowdsourcing tasks according to the problem attributes they can solve.

According to Nickerson and Zenger (2004), technological knowledge problems can be categorized based on two attributes: complexity and decomposability. *Complexity* measures the number of relevant knowledge domains and the intensity of their interactions for the technological problem under consideration. A simple problem involves very few knowledge domains, and the degree of interdependencies among them is low. Conversely, a complex problem involves a large number of relevant knowledge domains, with a high in-between degree of interactions. *Decomposability* measures the extent to which the problem under consideration can be divided into sub-problems. A decomposable problem can be divided into sub-problems each drawing on rather specialized and

Table 1. Taxonomy of Crowdsourcing Task Models According to the Problem Attributes They Can Handle

	Crowdsourcing task example	Attributes of problem to solve		
		Complex	Decomposable	Well-structured
Microtasks	Data categorization, curation, enrichment		✘	✘
Macrotasks	Taxonomy creation, itinerary planning, educational material development	✘	✘	✘
	"Holy grail", wicked problem solving	✘		
	Define a research methodology, formulate an R&D approach, refine system design	✘		✘

Each row is a task model. Our proposed task model is highlighted.

distinct knowledge sets so that it can be solved independently. Conversely, a non-decomposable problem cannot be subdivided as the interactions among knowledge domains are so extensive that it is virtually impractical to define sub-problems. For such a problem, if a solution is to be found, then it has to be an overall solution. Huang and Holden (2016), add a third important problem attribute, which is *structure*. A well-structured problem is one with clear boundaries of relevant knowledge domains, the interactions among which can be thoroughly understood; consequently, there are explicit and widely accepted approaches to solve the problem. Conversely, in ill-structured problems, the boundary of relevant knowledge domains is ambiguous and the interactions among them are poorly understood; thus no consensus approach can solve the problem.

This classification allows us to position the problems that current crowdsourcing approaches solve, the problems that the model proposed in this article intends to solve and the problems for which it is not suitable. Microtasks like data categorization, curation, or enrichment (Kittur et al. 2008; Musthag and Ganesan 2013), which form the focus of most commercial crowdsourcing applications, are intended to solve problems that are non-complex, well-structured, and decomposable. Macrotasks are almost always complex in that they require multiple interconnected knowledge domains, and they may be either decomposable or non-decomposable (e.g., open-ended work can be decomposable or not) and well- or ill-structured. The macrotasks that current crowdsourcing literature typically works on (Retelny et al. 2014) are complex, decomposable, and well structured. Examples include the following: creating a taxonomy (Chilton et al. 2013), planning an itinerary (Zhang et al. 2012), editing and correcting a document (Bernstein et al. 2010), or performing information tasks such as personal photography archive organization (Teevan et al. 2014). The problems that this work is intended to handle are also complex and well structured but they are non-decomposable. Examples include defining a research methodology or formulating an R&D approach. In general, these are problems often found at the beginning of an innovation project (when the broad objectives and solution criteria need to be set) and they are currently only processed manually, even if the rest of the project can be crowdsourced (Sieg et al. 2010). These problems can profit from a "continuity of useful action" (Altshuller 2005), where each consecutive contributor

maintains the global context and full semantic overview of the problem while iteratively refining it. Another example is drafting documents related to continuous process improvement (Anand et al. 2009) or to the refinement of a system's design until an acceptable solution is found, both of which necessitate the authors to maintain the overall context in making their contribution.

In the rest of this article, we adopt the term *macrotask* to describe complex work, which is not necessarily decomposable but that is aligned with the original macrotask definitions in that it requires more worker time (as in Cheng et al. (2015) and Haas et al. (2015)), can accept free-form (i.e. of continuous quality) worker inputs, and its quality can be determined through subjective evaluation, for example, peer review (as in Haas et al. (2015)).

Finally, our model is not suitable for ill-structured tasks, for which the interactions among the relevant knowledge domains (or even the exact required knowledge domains themselves), are not well understood. Such problems are also referred to as "holy grail" or "wicked" problems. Ill-structured problems are better served by open innovation idea contests (Majchrzak and Malhotra 2013), like Innovation Jam's discussion forums or Lego Mindstorms, where the purpose is to collect as many ideas as possible in search for the few breakthrough ideas rather than an iterative idea development. Processing ill-structured complex tasks with our model's sequential process could lead to problems such as fixation with one solution (Jansson and Smith 1991) or solution confounding (Little et al. 2010).

Summarizing, crowdsourcing optimization studies have so far extensively studied the assignment but not the sequencing aspect of the problem. This focus is to be expected, given their underlying assumption for tasks that can be decomposed to independent subtasks and processed in parallel by multiple workers. However, when decomposition is no longer an option, we need to split the timeline and sequence the assignments optimally across it in. In this work, we address the problem of optimal task sequencing for *non-decomposable, online, heterogeneous macrotasks*. In the next section, we will elaborate on the type of problems where non-decomposable tasks are necessary and how their optimization can have a significant impact on many contemporary platforms and applications.

3 TASK ASSIGNMENT AND SEQUENCING

In this section, we first describe from a high-level viewpoint the expert crowdsourcing setting that we target in this work (Section 3.1). Then we provide an example to illustrate the importance of adding the timeline sequencing element into the above setting (Section 3.2). Next, we define the TAS problem model, in terms of the input data, feasible solutions, constraints, and optimization goal (Section 3.3). Finally, we analyze this problem model in terms of complexity (Section 3.4).

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 settings:

(1) *Heterogeneous rather than homogeneous task pool.* We work with a crowdsourcing task pool that requires different skills and skill levels from the workers, and it includes multiple topics or knowledge domains (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 (where everyone can accomplish every task).
(2) *Non-decomposable rather than decomposable tasks.* Whereas decomposable tasks can be split to the level of independent subtasks and given simultaneously to multiple workers in a single assignment, non-decomposable tasks cannot be split and need to be processed in sequential assignments (one worker after the other).

(3) *Online rather than offline.* Rather than working on an offline setting, where the pool of workers and/or tasks are known *a priori*, 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 the task/worker information available up to the specific point in time.

Use Case. As an illustrative use case of a system where our model could be applied, imagine a corporate network of experts, in-house or across industries. The task that we will use is R&D problem and approach definition, which includes the formalization of a problem's scope, the outline of the research and development approach to be followed, the outline of the business plan, and so on. In essence, this task is a document that describes the needs of the innovation client and the scope and methodology of the R&D work to follow, and usually cannot be decomposed, because the authors need at this stage to maintain the full overview of dependencies across the affected areas (research, development, business, etc.) and the ability to change them. The definition of such a document typically takes place at the early stages of an innovation project, through iterative refinement cycles and with the participation of a few manually selected in-house experts. The correct and complete definition of this task is the backbone of any innovation project. Currently, such tasks cannot be crowdsourced, and their completeness relies on the in-house experts that can be found at the time of the task's definition (Sieg et al. 2010). Only after the above task has been defined, can the innovation process move to production stages, which can be crowdsourced, because they consist of separate chunks of work (e.g., code development).

For our use case, imagine a collaborative document writing platform, similar to a corporate wiki (albeit one with individual-person locks per article as we will see below). A person (innovation client) adds an article with a draft first description of the R&D problem that needs to be elaborated, and this article corresponds to a task in the terminology of our model. The innovation client also adds the task's knowledge domain and a quality threshold (e.g., in a scale from 0 to 10). Quality in this case can denote the level of completeness of the R&D problem description and the quality of the outline of the solution approach. Multiple metrics of quality can be used depending on the particular case and on what the R&D document describes, but for the sake of this use case, we will assume that they can be merged to a single numerical quality value.

As soon as the task is inserted in the system, the TAS algorithm searches for an expert worker, with an availability to undertake the task. Experts in this setting can denote their availability in advance, for example, in days (time slots in our model), and for the time period of the coming month (the timeline). Denoting availabilities can be important, since expertise in an innovation setting is a scarce resource: The experts can be professionals with demanding work schedules and less easy to replace than amateur workers in typical microtasking crowdsourcing settings. The selected expert works on the task, completing the definition of the R&D approach, filling missing points, and enriching the document with his/her viewpoint and expertise.

At the end of the day the article is locked, and its quality is assessed. Note that in the scope of the present work, we view the quality assessment mechanism as a black box to focus on the assignment and sequencing problem. In practice, task quality assessment can be implemented using a peer review process, with reviewers from the same expert worker network. Further, note that in our article we consider the expertise of each worker as a known numerical value. In practice, expertise can be calculated from the contribution history of each expert, i.e., the ratings that his/her work has received in the past through the same peer review process on other tasks of the same knowledge domain. Using these ratings, expertise can be calculated as the *average added quality* that the worker's contributions have given to articles of the same domain *in a single time step*. A description of the above mechanisms can be found in Lykourentzou et al. (2010).

If the task quality is below the defined threshold, then the algorithm selects another expert who will work on the task, contributing his/her expertise to improve the article's quality. Some workers with high expertise (e.g., 9 on a scale of 10) can improve the document substantially. Other workers, with lower expertise (e.g., 0.5 on a scale of 10), can only make minimal additional improvements (e.g., making a more strict formulation of an existing R&D hypothesis). This process, according to which expertise is accumulated to improve the innovation document iteration after iteration, corresponds to the *additive quality model* used in the objective function later on in our article.

3.2 The Importance of Worker Sequencing over Time Slots

Before giving the formal definition of the TAS optimization problem, we illustrate through an example the importance of looking at multiple timeslots and how this addition has an important effect on expert crowdsourcing performance.

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 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 additively and a required wage w that consumes this job's budget. Let us assume that three workers

$$i_0 = (2, 3), i_1 = (3, 2) \text{ and } i_2 = (2, 1)$$

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

$$j_0 : \{i_0, i_1\} \text{ or } \{i_1, i_2\}$$
$$j_1 : \{i_0, i_2\} \text{ or } \{i_1, i_2\}.$$

For both jobs, the second assignment seems to be preferable over the first, since workers $\{i_1, i_2\}$ provide more quality (= sum of the expertise of the assigned workers) for less cost. Now we look at a sequencing period of three timeslots and assume that only one worker may work on a job and at most one job is assigned to each worker per timeslot. Moreover, workers have limited availabilities 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 its 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 (i.e., that a job is processed by a sequence of workers and one worker at a time, as detailed by the *Sequentiality* model property in Section 3.3) with regard to this job. On the other hand, if we set i_2 on j_0 at timeslot 2, then 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 which timeslot is used for i_2 (*end of example*).

This 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 the appropriate constraints. We refer to this problem model as 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 it 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 referred to as *workers*, participating in the crowdsourcing platform. Each worker $i \in U$ has the following characteristics[4]:

— *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 remuneration that the worker requires 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.

[4]Note that in the context of this work, the quantification of worker expertise, wage, or speed are considered orthogonal to the studied assignment and sequencing problem. The interested reader is referred to (Lykourentzou et al. 2010; Dalip et al. 2011; Ipeirotis and Gabrilovich 2014) for available worker profile quantification techniques based on machine-learning, implicit evaluation or information theory.

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 reach.
- —*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 unless it reaches its quality threshold or it runs out of budget).

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 to at most a single task at a time. Sequentiality is chosen for three reasons. First, it is often imposed by the nature of expert non-decomposable crowdsourcing macrotasks, which do not allow multiple simultaneous worker contributions. 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 they have finished working on a task. Nevertheless, as also discussed in Section 6, an extension of our model to include worker concurrency is also feasible and can be examined as part of future work.

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 which 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 previous example in the present notation is

$$U_0 = (none, i_1, i_2)$$
$$U_1 = (i_2, none, i_0).$$

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

(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 they are 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 processed 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 $U_{jd} = \textit{none}$ for all j, d.

Objective. 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 with regard to this solution as $q_j = \sum_{i \in U_j} e_{ik}$, i.e., using an *additive quality model*.

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 with regard to 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, then 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 the worker expertise values 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 (Kellerer et al. 2013). The difference from 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 that 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. We can understand this partial problem as a machine-scheduling problem: Here workers play the role of machines, and jobs need to be processed 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 respective worker is assigned to this job or not). This aspect of TAS is a UNITTIME OPENSHOP problem with limited machine availability and job release dates (Kravchenko 2000). Note that the adoption of this model also implies non-preemption, i.e., a worker cannot be interrupted once they have 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 global deadline (which also marks the end of the timeline). Note also that the sequencing of an already fixed worker-task-assignment can be reduced to the BIPARTITE LIST EDGE-COLORING problem (Even et al. 1975). Here jobs and workers form a bipartite graph, where the worker-task-assignments are the graph's edges and timeslots are represented by colors. Subsequently, 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 described above are NP-hard on their own, and so is TAS, as we illustrate 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 3.1. *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 (Karp 1972). For finite, disjoint sets X, Y, and Z, we say that $M \subseteq X \times Y \times Z$ is a three-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 three-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, respectively) 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 $|W|$ 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 its 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 three-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 three-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 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 with regard to 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 Borodin and El-Yaniv (2005).

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, then 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 its 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 $O((|J'| + |U'|)^2|E|)$ (Kuhn 1955). So we obtain the following online algorithm (Algorithm 1: TAS-ONLINE) for TAS with polynomial running time $O(t|J|^3|U|^3)$.

This algorithm can be viewed as an online schema that allows multiple extensions, which we discuss in Section 6 on the basis of the present basic version.

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 the literature that no competitive algorithm for the *online* version of KNAPSACK exists where items arrive one at a time (Marchetti-Spaccamela and Vercellis 1995). An online algorithm is called competitive if the ratio of its performance and an optimal offline

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** $\{j \mapsto U_j \mid j \in J\}$

algorithm's performance can be bounded, a usual performance measure for online algorithms (Borodin and El-Yaniv 2005). It follows that no competitive online algorithm for TAS exists either. Therefore, to evaluate TAS-ONLINE experimentally, we formulate alternative algorithms to compare with (Section 5.1). We then conduct two types of experiments: synthetic (Sections 5.2, 5.3, and 5.4), where we experiment with a known simulated crowdsourcing instance and its variations, and real-world (Section 5.5), where we examine our model on an actual crowdsourcing platform. To evaluate our approach, we compare the algorithms principally in terms of the value of the objective function (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.

5.1 Alternative Algorithms

We evaluate the performance of TAS-ONLINE using five benchmarks, with each benchmark extending the previous with a new functionality. The first version of the algorithm (Algorithm 2: RANDOM) builds a feasible solution randomly and without any individual preferences of workers, similarly to a fully self-organized system. We simply iterate over the available workers and pick a feasible job.

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 (Algorithm 3: RANDOM EGOISTIC), we assume that workers try to pick the jobs offering a larger wage first, thus modeling a typical crowdsourcing environment, where workers are self-appointed to tasks trying to maximize their individual profit (Rogstadius et al. 2011). To do so, we substitute line 8 in the previous algorithm with the lines stated in Algorithm 3.

In the next step (Algorithm 4: RANDOM EGOISTIC FILTER), we extend RANDOM EGOISTIC with a filter that restricts the jobs that are offered to each worker based on expertise. This models the

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** $\{j \mapsto U_j \mid j \in J\}$

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**

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) (Downs et al. 2010; Jøsang et al. 2007), which allows us 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 a next variation of Algorithm 2, we design Algorithm 5: ONLINE GREEDY QUALITY. This algorithm uses an adaptation of a known strategy for online task assignment (Ho and Vaughan 2012), in which a job is chosen for a worker 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 (and leaving line 7 unchanged).

ALGORITHM 5: ONLINE GREEDY QUALITY

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

Finally, as a last variation we design Algorithm 6: ONLINE GREEDY EFFICIENCY. This algorithm changes the metric for job selection used by Algorithm 5: ONLINE GREEDY QUALITY to the best ratio of expertise per wage unit (efficiency), i.e., it uses the same profit function used in TAS-ONLINE.

ALGORITHM 6: ONLINE GREEDY EFFICIENCY

80 *pick job* $j \in J'_i$ *such that* e_{ik_j}/w_{ik_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 7: TAS-OFFLINE) that does not have to proceed day-by-day but has access to the complete input at once. This clairvoyant algorithm knows in advance which workers will be available on which days of the scheduling period and also which 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 with regard to 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 $[r_j, r_j + lookahead]$ from which the available workers are chosen to control the maximum flow time of each job. *Flow time* for job j is the number of timeslots between the job's release date r_j and the latest worker assignment on j. Second, we use *minavail* to ensure that each worker has at least *minavail*-many free timeslots remaining in the above interval to facilitate the allocation of timeslots afterwards.

ALGORITHM 7: TAS-OFFLINE

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** $j \in J$ **do** // ordered by release dates
3 U'_j = *feasible workers for* j
4 **for** $i \in U'_j$ **do**
5 **if** ***not****minavail in* $[r_j, r_j + lookahead]$ **then**
6 *remove* i *from* U'_j
7 $W_j \leftarrow DynProgKnapsack(U'_j, Q_j, C_j)$
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

Table 2. Objective Function (Completed Jobs)

Algorithm	absolute	% of upper bound
RANDOM	2	0, 39
RANDOM EGOISTIC	98	19, 03
RANDOM EGOISTIC FILTER	114	22, 14
ONLINE GREEDY QUALITY	82	15, 92
ONLINE GREEDY EFFICIENCY	112	21, 75
TAS-ONLINE	**355**	**68, 93**
TAS-OFFLINE	411	79, 81

time algorithm (the DP-table has dimension $|U| \times C_j$ for each job) (Kellerer et al. 2013). 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.

5.2 Synthetic Data Experiment

We first experiment with synthetic data, which were generated using the experimental result distributions reported in Basu Roy et al. (2015), 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 1, 000 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

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 of the 600 jobs can be completed.

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 than the other online algorithms (cf. Table 2). Note that in Tables 2–4, we highlight the values of TAS-ONLINE, since these need to be compared with the other entries.

To get a more precise picture, we not only want to compare these algorithms with regard to 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 (cf. Table 3). In both cases, we take the average values

Table 3. Number of Assigned Workers and Flow Time

Algorithm	assigned workers	flow time
RANDOM	4, 77	4, 77
RANDOM EGOISTIC	3, 46	3, 46
RANDOM EGOISTIC FILTER	1, 64	7, 37
ONLINE GREEDY QUALITY	3, 56	2, 91
ONLINE GREEDY EFFICIENCY	3, 48	3, 02
TAS-ONLINE	**3, 31**	**3, 31**
TAS-OFFLINE	2, 41	8, 11

Table 4. Budget Usage and Reached Quality in %

Algorithm	used budget	reached quality
RANDOM	88, 16	60, 62
RANDOM EGOISTIC	92, 05	90, 27
RANDOM EGOISTIC FILTER	52, 6	55, 77
ONLINE GREEDY QUALITY	91, 44	87, 12
ONLINE GREEDY EFFICIENCY	92	90, 26
TAS-ONLINE	**94, 35**	**97, 76**
TAS-OFFLINE	67, 73	70, 21

over all jobs in the system (not only the completed ones). 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 look at how much budget is used with these assignments and how much quality is reached, both relatively to the given thresholds and on average over all the jobs in the system (cf. Table 4). Due to its greedy nature TAS-ONLINE reaches very high quality values, not only regarding complete but also incomplete jobs and it exploits the available budgets to a large extent.

Finally, we examine how the main performance measures develop over time, in terms of the completed jobs (Figure 2) and in terms of the quality reached (Figure 3). Interestingly, we observe that the higher values in Figure 2 for TAS-OFFLINE appear towards the end of the scheduling period. A possible explanation is that the lookahead mechanism of this algorithm takes the end of the timeline into account.

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 as follows: 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 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 4–13. 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.

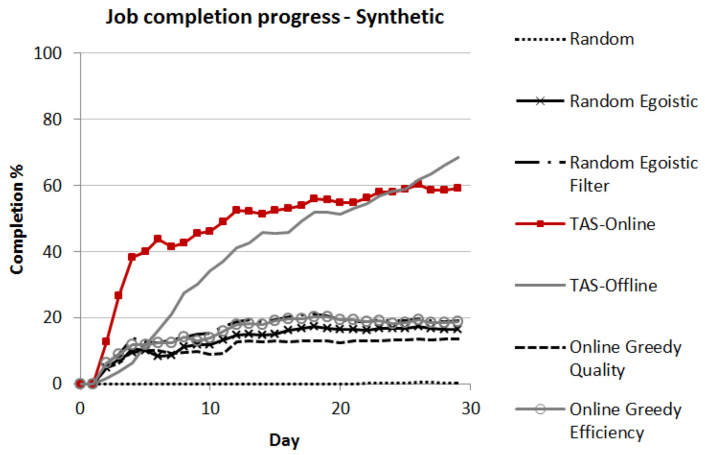

Fig. 2. Job completion over time for each of the five tested algorithms. The proposed algorithm TAS-ONLINE manages to achieve more completed jobs most of the time compared to its competitors. Time unit expressed in days.

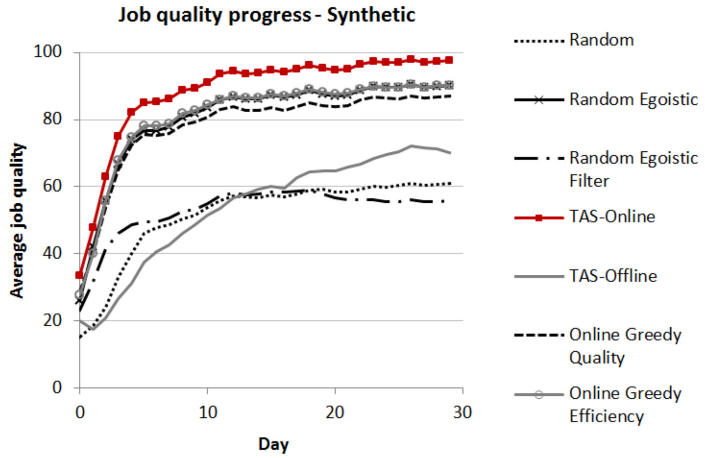

Fig. 3. Average job quality over time for each of the five tested algorithms. The proposed algorithm TAS-ONLINE manages to achieve higher quality per time unit compared to its competitors. Time unit expressed in days.

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 examined 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.

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 parameters. This result indicates

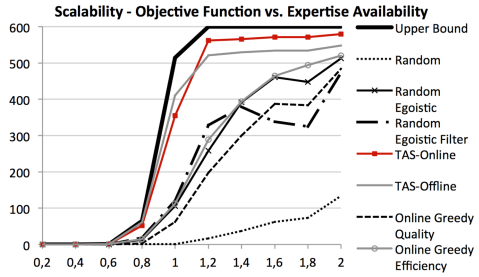

Fig. 4. Objective function vs. expertise availability. The vertical line corresponds to the baseline simulated instance.

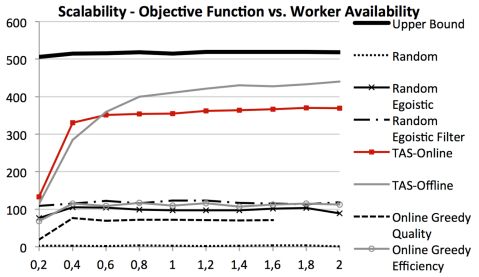

Fig. 5. Objective function vs. worker availability.

that the performance of the algorithm as detailed in Section 5.2 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 4, 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 those of the baseline instance TAS-OFFLINE is the best-performing algorithm, followed closely by TAS-ONLINE, while for expertise levels slightly higher than those of the baseline TAS-ONLINE takes and maintains precedence. This can be attributed to the fact that when worker expertise is low, the offline version performs better, since it knows worker availabilities beforehand and can thus calculate better assignments. In case of high expertise, the look into the future done by the offline algorithm seems to be of no particular advantage, since a sufficiently high number of good assignments are possible at any given moment. The fact that the online does even better than the onffline can be explained due to the different design patterns of the two algorithms (recall that the offline is not optimal, since an optimal algorithm cannot be applied for instances of this size within reasonable runtime unless P=NP). Moreover, 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 5). 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 (Figures 7, 9, 11, and 13).

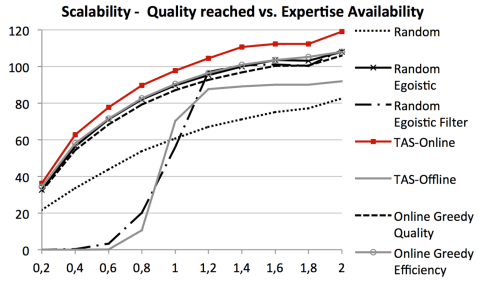

Fig. 6. Quality reached vs. expertise availability.

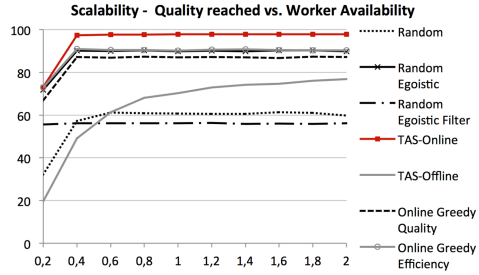

Fig. 7. Quality reached vs. worker availability.

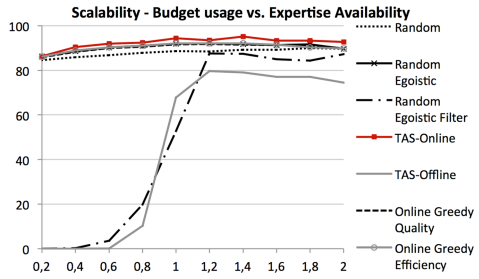

Fig. 8. Budget usage vs. expertise availability.

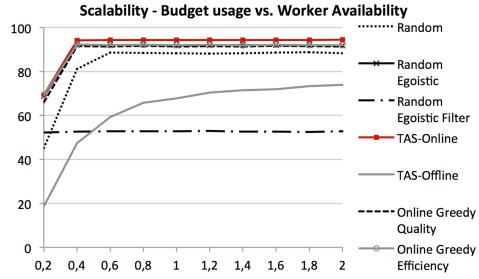

Fig. 9. Budget usage vs. worker availability.

5.3.3 Scalability Effect on Quality: TAS-ONLINE *Achieves Higher Quality.* Figure 6 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 both versions of ONLINE GREEDY are the next best performing algorithms, respectively, but, unlike TAS-ONLINE, they achieve their high-quality results at the cost of accomplishing too few jobs, as can be seen by juxtaposing Figures 4 and 6 (similarly for Figures 5 and 7).

5.3.4 Scalability on Other Parameters: TAS-ONLINE *Performs Comparably to 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 8, smaller y-axis values are better). As we may observe, TAS-ONLINE consumes most (≈90%) of its available budget at the same consumption level as the ONLINE GREEDY QUALITY, ONLINE GREEDY EFFICIENCY, 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 4), where TAS-ONLINE consistently accomplishes more jobs (almost up to double for $x = 1.2$) than RANDOM EGOISTIC FILTER. Similar results can be observed when scaling worker availability (Figures 9 and 5).

Effect on Flow Time. Figure 10 shows the flow time of the algorithms for varying levels of expertise availability, and smaller y-axis values are better. As we can observe, TAS-ONLINE behaves similarly to the rest of the online algorithms. This shows that there is no tradeoff of performance

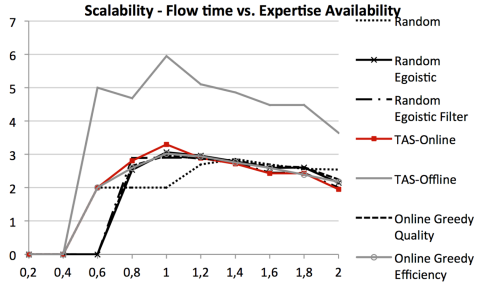

Fig. 10. Flow time vs. expertise availability.

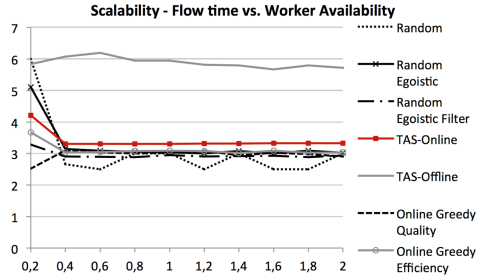

Fig. 11. Flow time vs. worker availability.

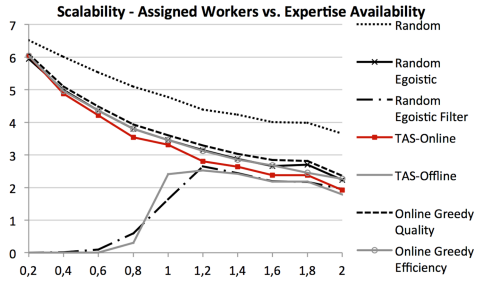

Fig. 12. Number of assigned workers vs. expertise availability.

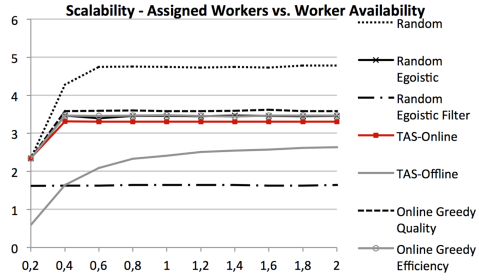

Fig. 13. Number of assigned workers vs. worker availability.

for time, i.e., the proposed algorithm does not achieve its higher objective function values at the cost of flow time. Similar results can be observed when scaling worker availability (Figure 11).

Effect on Number of Assigned Workers. Figure 12 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, since the algorithms need to assign multiple workers to achieve the quality thresholds when expertise is scarce. Similar results can be observed when scaling worker availability (Figure 13).

5.4 Job Prioritization

During the scheduling period, some jobs may become more urgent than others, for example, in crowdsourcing systems dealing with crisis response (Imran et al. 2013) or multi-tenant crowd systems where concurrent requesters can impose different job priorities (Difallah et al. 2016). Job prioritization can be incorporated in the model based on flow time, 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 in each time slot for a close progress control. For instance, makespan prioritization can be used when worker availability is low and removed when more workers become available. To point out the applicability of TAS-ONLINE in these situations, we have conducted an experiment with an adapted version of our algorithm, where jobs with currently larger flow time have strictly

Table 5. TAS-ONLINE with and without Priorities

Algorithm	completed jobs	flow time
TAS-ONLINE	131	4, 19
TAS-ONLINE + priorities	173	3, 18

higher profits over jobs with smaller flow time. It turns out that this prioritization has a notable effect when worker availability is low (scaling factor 0.2), as can be observed from Table 5.

So the number of completed jobs is larger and flow time is smaller with priorities. This effect vanishes when the scaling factor for worker availability increases. Certainly, introducing priorities does not guarantee the absence of long-running jobs or even starvation. To deal with these situations, a monitoring procedure together with a stopping mechanism according to some policy could be added easily to TAS-ONLINE. Note that all other algorithms in our experiments would have to utilize such a mechanism much more frequently due to more incomplete jobs.

5.5 Real Data Experiment

In addition to the synthetic experiments, we conducted a small-scale real-world experiment to test the applicability of the proposed model on real workers and tasks. In contrast to the synthetic experiments, this experiment is intended to provide an initial and qualitative viewpoint of the model's performance in a real-world setting. The platform we used was CrowdFlower.[5] 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, enriching a news article text on a given topic. 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 (i.e., work for approximately 30 min on one particular day of the next 8 days, as explained in the algorithm setup of the timeline parameter later) and asked them to provide us with their required wage and availability per day. Worker expertise was on average 5.02 (std=1.86) for the 'The FIFA 2015 corruption scandal" and 4.78 (std=2.11) for the "Self-driving cars" domain. The requested wage on average was \$4.9 (std=\$2.7). Worker availability was on average 20 workers per day (std= 8.12). Note that, to keep up with the requirements of the online setting, only the availability of each given day was considered known by the algorithms and used only to invite the selected (among the available) workers, i.e., to avoid spamming the rest (who had declared to be unavailable on that day).

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. To keep up with the real worker wage requirements, the job cost threshold was adjusted accordingly, with 100 cost points of the synthetic experiment corresponding to \$10 in the real data experiment.

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

In regards to the algorithms to be compared we used RANDOM EGO. FILTER as benchmark with *factor* = 0.3 (a worker was allowed to undertake a job only if their expertise was at least 30% of the target job quality) and TAS-ONLINE as the optimization algorithm. This algorithm was used as the benchmark given that it had the best performance among the online competitors of TAS-ONLINE in the synthetic experiments (see Table 2). 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. The writing instructions that the document contained for the worker were as follows (using as an example the "FIFA 2015 corruption scandal" knowledge domain):

"Welcome! In the previous round you passed a multiple choice test about your knowledge on the FIFA 2015 scandal. Here you will write a short news article about this topic. Assume that you are a journalist and that you have been assigned to write or modify a news article about the recent FIFA 2015 corruption scandal. After your contribution the article must (i) be approximately 150 words in length, (ii) have a clear descriptive title, (iii) cover the specific news event as completely and accurately as possible, (iv) be unique (do not copy from another news source), and (v) be grammatically and syntactically correct."

After these instructions and in the same document, the worker found either a blank article page to complete (if the article did not yet exist), or another worker's previous version of the article, which they would need to modify. In the same instructions page, workers were reminded that they had one day to finish the task. At the end of that day, the document would be locked for the particular worker and sent for evaluation to a crowd of 50 independent crowd workers (different than those used in the experiments) to evaluate the job's current quality. Each of the 50 workers evaluated each document, on a 5-point Likert scale across the same five quality dimensions of the instructions given to the author workers (i.e., length, title clarity, news coverage completeness, uniqueness, and grammatical/syntactical correctness), and the documents were given to them in a round-robin fashion. The five quality dimensions were averaged to a single numerical value per document, and this value was used to measure the document's current quality. In case the document 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 three of six jobs, while the optimization algorithm achieved successful completion of five of six jobs. As expected, the benchmark algorithm either allowed workers of the minimum necessary expertise to take a job, thus delaying the job's 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 14 and 15, similarly to the respective results of the synthetic experiments, TAS-ONLINE achieved higher average job quality and job completion percentages than the benchmark.

6 DISCUSSION AND FUTURE EXTENSIONS

The TAS model presented in this article is the first concrete attempt to incorporate sequencing in the optimization of non-decomposable expert crowdsourcing macrotasks. 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 section, we briefly discuss how the proposed TAS model and the TAS-ONLINE algorithm can be adapted to address further challenging settings 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 to recruit better qualified workers

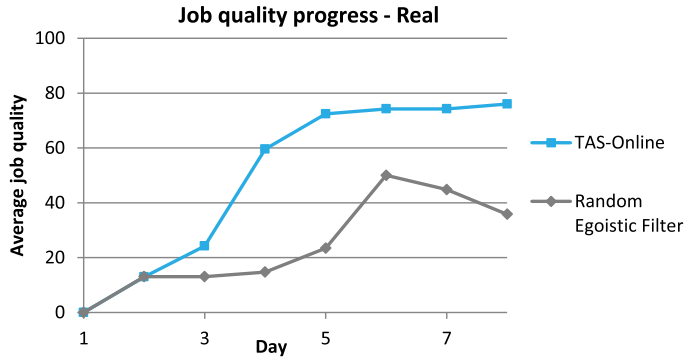

Fig. 14. Average job quality over time. The proposed algorithm TAS-ONLINE manages to achieve higher quality per time unit compared to the benchmark. Time unit expressed in days.

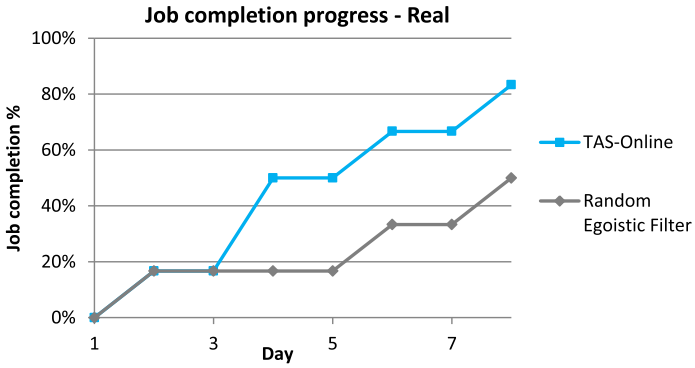

Fig. 15. Job completion percentage over time. The proposed algorithm TAS-ONLINE manages to achieve higher job completion rates per time unit compared to the benchmark. Time unit expressed in days.

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 tradeoffs.

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 setting 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.

Quality Aggregation Model. In this article, we model task quality by summing the expertise of the task contributors, where expertise is assumed to be the added quality that a worker can bring

to the tasks of a given domain. This quality model is meant to capture the iterative improvements, sometimes important and sometimes minimal, which multiple workers, one after the other, can produce thanks to their different perspectives and knowledge. Experimental grounding for the additive quality model can be found in prior literature. Little et al. (2010) show that crowd workers iterating on one another's contributions produced tasks of increasing average quality, with statistically significant experimental results for writing and brainstorming tasks. Dai et al. (2010) propose an iterative workflow, overseen by a decision theory-based automated planner, which consists of incremental task improvements followed by quality evaluations through worker voting until a certain task quality level is reached. Results show that this iterative model can yield higher task quality compared to non-iterative processes. Roy et al. (2015, 2014) integrate the additive task quality model in a framework for computing task assignments based on pre-indexing and show that it can help improve the quality of knowledge-intensive tasks like news writing. Finally, Goto et al. (2016) extend the additive model with a penalty in case a worker's contribution fails to improve the task.

In the future, other models for calculating task quality could be considered, such as the maximum contribution one, where task quality is determined by the worker with the highest expertise among those that have contributed to the task. Our algorithm could be adapted to accommodate these models, thanks to the flexible graph structure that it uses to represent the possible assignments of workers and tasks. Essentially, the algorithm matches one numerical property of the particular worker-task-relation (in this article the profit calculated as the worker's expertise in the domain of the task divided by their wage for that domain) using the edges of a bipartite graph, which initially connects each task to each worker and then is reduced according to the constraints of the given problem model.

Adaptations to the way that the edges are dropped can allow implementing different quality evaluation models. For example, in this article a worker-task edge is dropped when (i) the worker has already contributed to the task, (ii) the worker has no expertise in the task's domain, or (iii) the wage of the worker for this domain is greater than the remaining budget of the task. In case the maximum quality calculation model is used, meaning that a new worker can only be matched to a task if she brings a quality higher than any of the previously allocated workers, then the algorithm should also drop all worker-task edges pointing to workers with expertise less than the quality currently achieved for the specific task. As a consequence, the algorithm would return feasible assignments of workers to tasks such that for each single task only workers with increasing expertise are allocated over time, while respecting budget limitations and other constraints.

Adaptations to the graph edge values could also be foreseen and lead to further interesting future research. For example, in this article, a worker's expertise is calculated per domain, i.e., it is considered identical for all tasks of that domain for the particular worker. By changing the values of selected worker-task edges, the algorithm can be adapted to handle per task rather than per-domain expertise and quality models. Or even more specifically, worker expertise could also be calculated taking into account the past improvements of the individual task, meaning that consecutive worker contributions will have a decreasing marginal added quality when the already-achieved task quality increases. The algorithm could be customized to address this expertise calculation mode by changing and recalculating the worker-task edge values per time slot. But although the algorithm basis (i.e. the graph) can be easily adjusted to accommodate the maximum quality and other calculation models or problem requirements, adopting these models raises the need for further research and experimental evaluation. We believe that one interesting research direction in this context could be looking for mechanisms to postpone assignments based on worker arrival predictions to treat the given budget with care.

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" (Dow et al. 2012; Basu Roy et al. 2013). An improved version of the model could recognize this 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 Lykourentzou et al. (2010)). The online version of the TAS algorithm is particularly well suited for such a 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, for example, 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. Constraint (a) of our examined TAS model allows only one task per worker and timeslot. Note that the model is in fact more general, since one can introduce fake workers for the same person and hence generate multiple assignments per worker per timeslot. Furthermore, constraint (c) demands that a worker can only be assigned once to the same job along the timeline. This is a simplified model assumption. In case one considers more than one timeslot per worker (with or without preemption), this would result in a much more complex model. For example, this would induce the need to calculate and adjust a worker's expertise depending on the number of contributions of this worker the same job, because the "amount" of his/her contribution will not be same in a second assignment as opposed to a first one (e.g., due to learning effects, where the worker is already familiar with the job). Certainly, this can be further studied in future work.

Algorithm Extensions Using Lookahead. The algorithm proposed in this article is a first attempt to solve the problem of task assignment optimization for non-decomposable macrotasks. At each given time slot, the algorithm calculates a feasible matching of workers to tasks for the next time slot, and it does so in an online fashion, i.e., without depending on any future information about the workers or the tasks. This process results in having calculated, at each time slot, a feasible sequence of workers per task up to that slot. Nevertheless, and although the problem model involves worker sequencing, the algorithm does not calculate future sequences of workers spanning multiple future slots. As a future extension the algorithm could be coupled with a lookahead mechanism, according to which it could rely on past worker and task information to foresee future worker availability, worker performance, and task arrivals and then use these predictions to compute worker sequences per task spanning multiple slots. Adding lookahead could improve the algorithm's performance, for example, in terms of computation time: Instead of revisiting its assignment decisions at every single time slot to adapt to the dynamic conditions of the crowdsourcing setting (changes in worker availability or task arrival rates), lookahead could enable the algorithm to plan ahead for multiple slots and run only periodically to verify the plan or make marginal adaptations. Naturally, this would give rise to future research, necessary to explore the depth of lookahead that could provide the best tradeoff between computation time and efficient 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 types of jobs at hand. As such, they could be used independently or in various combinations as the starting points for further studies in expert crowdsourcing optimization.

7 CONCLUSION

In this article, we present TAS, a problem model that examines online optimization in expert crowdsourcing settings that involve non-decomposable macrotasks. We prove the problem's NP-hardness and propose a greedy assignment and sequencing algorithm, namely TAS-ONLINE, to address it. We illustrate through simulated and real-world experiments that optimization under this model can significantly improve performance. Our results have implications for enhancing the Quality of Service of crowdsourcing platforms offering non-decomposable complex tasks, but also for allowing online expert crowdsourcing communities to make better use of their human capital and available expertise. Multiple future extensions can be foreseen. These include extending the proposed TAS model to handle requirements such as budget flexibility, the non-acceptance of assignments by the workers, different job quality aggregation mechanism, learning, varying modes of assignment order and number of assignments per worker, as well as forecasting.

ACKNOWLEGMENTS

The authors would like to thank the anonymous reviewers for their constructive comments.

REFERENCES

Genrich Altshuller. 2005. *The Innovation Algorithm: TRIZ, Systematic Innovation and Technical Creativity*. Worchester, MA: Technical Innovation Center.

Vamshi Ambati, Stephan Vogel, and Jaime Carbonell. 2012. Collaborative workflow for crowdsourcing translation. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW'12)*. ACM, New York, NY, 1191–1194. DOI: http://dx.doi.org/10.1145/2145204.2145382

Gopesh Anand, Peter T. Ward, Mohan V. Tatikonda, and David A. Schilling. 2009. Dynamic capabilities through continuous improvement infrastructure. *J. Operat. Manage.* 27, 6 (2009), 444–461. DOI: http://dx.doi.org/10.1016/j.jom.2009.02.002

Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. 2013. Crowds, not drones: Modeling human factors in interactive crowdsourcing. In *Proceedings of the VLDB Workshop on Databases and Crowdsourcing (DBCrowd'13)*, Reynold Cheng, Anish Das Sarma, Silviu Maniu, and Pierre Senellart (Eds.). 39–42.

Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. 2014. Optimization in knowledge-intensive crowdsourcing. *ArXiv e-prints*. http://adsabs.harvard.edu/abs/2014arXiv1401.1302B.

Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. 2015. Task assignment optimization in knowledge-intensive crowdsourcing. *The VLDB Journal* 24, 4 (2015), 467–491. DOI: http://dx.doi.org/10.1007/s00778-015-0385-2

Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. 2011. 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)*. ACM, New York, NY, 33–42. DOI: http://dx.doi.org/10.1145/2047196.2047201

Michael S. Bernstein, David R. Karger, Robert C. Miller, and Joel Brandt. 2012. Analytic methods for optimizing realtime crowdsourcing. *CoRR* abs/1204.2995 (2012). http://dblp.uni-trier.de/db/journals/corr/corr1204.html#abs-1204-2995.

Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: A word processor with a crowd inside. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology (UIST'10)*. ACM, New York, NY, 313–322. DOI: http://dx.doi.org/10.1145/1866029.1866078

Arpita Biswas, Shweta Jain, Debmalya Mandal, and Y. Narahari. 2015. 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)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1101–1109.

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

Ioannis Boutsis and Vana Kalogeraki. 2014. On task assignment for real-time reliable crowdsourcing. In *2014 IEEE 34th International Conference on Distributed Computing Systems (ICDCS'14)*. 1–10. DOI: http://dx.doi.org/10.1109/ICDCS.2014.9

Jonathan Bragg, Andrey Kolobov, Mausam, and Daniel S. Weld. 2014. Parallel task routing for crowdsourcing. In *Proceedings of the 2nd AAAI Conference on Human Computation and Crowdsourcing*.

Joel Chan, Steven Dang, and Steven P. Dow. 2016. Improving crowd innovation with expert facilitation. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW'16)*. ACM, New York, NY, 1223–1235. DOI: http://dx.doi.org/10.1145/2818048.2820023

Justin Cheng, Jaime Teevan, Shamsi T. Iqbal, and Michael S. Bernstein. 2015. Break it down: A comparison of macro- and microtasks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*. ACM, New York, NY, 4061–4064. DOI: http://dx.doi.org/10.1145/2702123.2702146

Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. 2013. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 1999–2008. DOI: http://dx.doi.org/10.1145/2470654.2466265

Peng Dai, Mausam Daniel, and S. Weld. 2010. Decision-theoretic control of crowd-sourced workflows. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI10)*.

Daniel Hasan Dalip, Marcos André Gonçalves, Marco Cristo, and Pável Calado. 2011. Automatic assessment of document quality in web collaborative digital libraries. *J. Data Inform. Qual.* 2, 3, Article 14 (Dec. 2011), 30 pages. DOI: http://dx.doi.org/10.1145/2063504.2063507

Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. 2016. Scheduling human intelligence tasks in multi-tenant crowd-powered systems. In *Proceedings of the 25th International Conference on World Wide Web (WWW'16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 855–865. DOI: http://dx.doi.org/10.1145/2872427.2883030

Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. 2012. Shepherding the crowd yields better work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW'12)*. ACM, New York, NY, 1013–1022. DOI: http://dx.doi.org/10.1145/2145204.2145355

Julie S. Downs, Mandy B. Holbrook, Steve Sheng, and Lorrie Faith Cranor. 2010. Are your participants gaming the system?: Screening mechanical turk workers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'10)*. ACM, New York, NY, 2399–2402. DOI: http://dx.doi.org/10.1145/1753326.1753688

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

Ju Fan, Meihui Zhang, Stanley Kok, Meiyu Lu, and Beng Chin Ooi. 2015. CrowdOp: Query optimization for declarative crowdsourcing systems. *IEEE Trans. Knowl. Data Eng.* 99 (2015), 1–1. DOI: http://dx.doi.org/10.1109/TKDE.2015.2407353

Siamak Faradani, Bjoern Hartmann, and Panagiotis G. Ipeirotis. 2011. What's the right price? Pricing tasks for finishing on time. In *Human Computation (AAAI Workshops)*, Vol. WS-11-11. AAAI.

Teppo Felin and Todd R. Zenger. 2014. Closed or open innovation? Problem solving and the governance choice. *Res. Policy* 43, 5 (2014), 914–925. DOI: http://dx.doi.org/10.1016/j.respol.2013.09.006 Open Innovation: New Insights and Evidence.

Gagan Goel, Afshin Nikzad, and Adish Singla. 2014. 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)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 279–280. DOI: http://dx.doi.org/10.1145/2567948.2577311

Shinsuke Goto, Toru Ishida, and Donghui Lin. 2016. Understanding crowdsourcingworkflow: Modeling and optimizing iterative and parallel processes. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing (HCOMP'16)*.

Daniel Haas, Jason Ansel, Lydia Gu, and Adam Marcus. 2015. Argonaut: Macrotask crowdsourcing for complex data processing. *Proc. VLDB Endow.* 8, 12 (Aug. 2015), 1642–1653. DOI: http://dx.doi.org/10.14778/2824032.2824062

Chien-Ju Ho and Jennifer Wortman Vaughan. 2012. Online task assignment in crowdsourcing markets. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*.

John Joseph Horton and Lydia B. Chilton. 2010. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM Conference on Electronic Commerce (EC'10)*. ACM, New York, NY, 209–218. DOI: http://dx.doi.org/10.1145/1807342.1807376

Shaopeng Huang and Darryl Holden. 2016. The R&D boundaries of the firm: A problem-solving perspective. *International Journal of the Economics of Business* 23, 3 (2016), 287–317. DOI: https://doi.org/10.1080/13571516.2016.1156247

Muhammad Imran, Ioanna Lykourentzou, Yannick Naudet, and Carlos Castillo. 2013. Engineering crowdsourced stream processing systems. *ArXiv e-prints*. http://adsabs.harvard.edu/abs/2013arXiv1310.5463I.

Panagiotis G. Ipeirotis and Evgeniy Gabrilovich. 2014. Quizz: Targeted crowdsourcing with a billion (potential) users. In *Proceedings of the 23rd International Conference on World Wide Web (WWW'14)*. ACM, New York, NY, 143–154. DOI: http://dx.doi.org/10.1145/2566486.2567988

David G. Jansson and Steven M. Smith. 1991. Design fixation. *Des. Stud.* 12, 1 (1991), 3–11. DOI: http://dx.doi.org/10.1016/0142-694X(91)90003-F

Audun Jøsang, Roslan Ismail, and Colin Boyd. 2007. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.* 43, 2 (Mar. 2007), 618–644. DOI: http://dx.doi.org/10.1016/j.dss.2005.05.019

Saraschandra Karanam, Deepthi Chander, L. Elisa Celis, Koustuv Dasgupta, and Vaibhav Rajan. 2014. Adaptive performance optimization over crowd labor channels. In *Proceedings of the 2nd AAAI Conference on Human Computation and Crowdsourcing (HCOMP'14)*.

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

Richard M. Karp. 1972. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*. Springer US, 85–103.

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

Roman Khazankin, Benjamin Satzger, and Schahram Dustdar. 2012. Optimized execution of business processes on crowdsourcing platforms. In *Proceedings of the 2012 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom'12)*. 443–451.

Joy Kim, Justin Cheng, and Michael S. Bernstein. 2014. Ensemble: Exploring complementary strengths of leaders and crowds in creative collaboration. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW'14)*. ACM, New York, NY, 745–755. DOI: http://dx.doi.org/10.1145/2531602.2531638

Joy Kim, Sarah Sterman, Allegra Argent Beal Cohen, and Michael S. Bernstein. 2017. Mechanical novel: Crowdsourcing complex work through reflection and revision. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW'17)*. ACM, 233–245. DOI: http://doi.acm.org.proxy.bnl.lu/10.1145/2998181.2998196

Aniket Kittur, Ed H. Chi, and Bongwon Suh. 2008. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'08)*. ACM, New York, NY, 453–456. DOI: http://dx.doi.org/10.1145/1357054.1357127

Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. 2011. CrowdForge: Crowdsourcing complex work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST'11)*. ACM, New York, NY, 43–52. DOI: http://dx.doi.org/10.1145/2047196.2047202

Svetlana A. Kravchenko. 2000. On the complexity of minimizing the number of late jobs in unit time open shop. *Discrete Applied Mathematics* 100, 1 (March 2000), 127–132. DOI: https://doi.org/10.1016/S0166-218X(99)00202-4

Harold W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2, 1–2 (1955), 83–97. DOI: http://dx.doi.org/10.1002/nav.3800020109

Anand Kulkarni, Matthew Can, and Björn Hartmann. 2012. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW'12)*. ACM, New York, NY, 1003–1012. DOI: http://dx.doi.org/10.1145/2145204.2145354

Walter Lasecki, Christopher Miller, Adam Sadilek, Andrew Abumoussa, Donato Borrello, Raja Kushalnagar, and Jeffrey Bigham. 2012. Real-time captioning by groups of non-experts. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST'12)*. ACM, New York, NY, 23–34. DOI: http://dx.doi.org/10.1145/2380116.2380122

Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. 2010. Exploring iterative and parallel human computation processes. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP'10)*. ACM, New York, NY, 68–76. DOI: http://dx.doi.org/10.1145/1837885.1837907

Ioanna Lykourentzou, Katerina Papadaki, Dimitrios J. Vergados, Despina Polemi, and Vassili Loumos. 2010. CorpWiki: A self-regulating wiki to promote corporate collective intelligence through expert peer matching. *Inf. Sci.* 180, 1 (2010), 18–38. DOI: http://dx.doi.org/10.1016/j.ins.2009.08.003 Special Issue on Collective Intelligence.

Ioanna Lykourentzou and Heinz Schmitz. 2015. An Online Approach to Task Assignment and Sequencing in Expert Crowdsourcing. *In Proceedings of the 3rd AAAI Conference on Human Computation and Crowdsourcing (HCOMP'15)* .

A. Majchrzak and A. Malhotra. 2013. Towards an information systems perspective and research agenda on crowdsourcing for innovation. *J. Strat. Inf. Syst.* 22, 4 (2013), 257–268. DOI: http://dx.doi.org/10.1016/j.jsis.2013.07.004

A. Marchetti-Spaccamela and C. Vercellis. 1995. Stochastic on-line knapsack problems. *Math. Program.* 68, 1–3 (Jan. 1995), 73–104.

Patrick Minder, Sven Seuken, Abraham Bernstein, and Mengia Zollinger. 2012. Crowdmanager-combinatorial allocation and pricing of crowdsourcing tasks with time constraints. In *Proceedings of the 2nd Workshop on Social Computing and User Generated Content*. ACM Conference on Electronic Commerce (ACM-EC 2012).

Luyi Mo, Reynold Cheng, Ben Kao, Xuan S. Yang, Chenghui Ren, Siyu Lei, David W. Cheung, and Eric Loz. 2013. Optimizing plurality for human intelligence tasks. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'13)*. ACM, New York, NY. DOI: http://dx.doi.org/10.1145/2505515.2505755

Mohamed Musthag and Deepak Ganesan. 2013. Labor dynamics in a mobile micro-task market. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 641–650. DOI: http://dx.doi.org/10.1145/2470654.2470745

Swaprava Nath, Pankaj Dayama, Dinesh Garg, Yadati Narahari, and James Zou. 2012. 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)*. Springer-Verlag, Berlin, 212–226. DOI: http://dx.doi.org/10.1007/978-3-642-35311-6_16

Jack A. Nickerson and Todd R. Zenger. 2004. A knowledge-based theory of the firm–the problem-solving perspective. *Org. Sci.* 15, 6 (Nov. 2004), 617–632. DOI: http://dx.doi.org/10.1287/orsc.1040.0093

Aditya Parameswaran, Akash Das Sarma, and Vipul Venkataraman. 2016. Optimizing Open-Ended Crowdsourcing: The Next Frontier in Crowdsourced Data Management. *ArXiv e-prints*. http://adsabs.harvard.edu/abs/2016arXiv161005377P.

Vaibhav Rajan, Sakyajit Bhattacharya, Elisa L. Celis, Deepthi Chander, Koustuv Dasgupta, and Sarachandra Karanam. 2013. CrowdControl: An online learning approach for optimal task scheduling in a dynamic crowd platform. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC'12)*.

Daniela Retelny, Sébastien Robaszkiewicz, Alexandra To, Walter S. Lasecki, Jay Patel, Negar Rahmati, Tulsee Doshi, Melissa Valentine, and Michael S. Bernstein. 2014. Expert crowdsourcing with flash teams. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST'14)*. ACM, New York, NY, 75–85. DOI: http://dx.doi.org/10.1145/2642918.2647409

Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. 2011. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In *Proceedings of the 5th International Conference on Weblogs and Social Media*. http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2778.

Jan Henrik Sieg, Martin W. Wallin, and Georg Von Krogh. 2010. Managerial challenges in open innovation: A study of innovation intermediation in the chemical industry. *R&D Manage.* 40, 3 (2010), 281–291. DOI: http://dx.doi.org/10.1111/j.1467-9310.2010.00596.x

Jaime Teevan, Shamsi T. Iqbal, Carrie J. Cai, Jeffrey P. Bigham, Michael S. Bernstein, and Elizabeth M. Gerber. 2016. Productivity decomposed: Getting big things done with microtasks. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA'16)*. ACM, New York, NY, 3500–3507. DOI: http://dx.doi.org/10.1145/2851581.2856480

Jaime Teevan, Daniel J. Liebling, and Walter S. Lasecki. 2014. Selfsourcing personal tasks. In *Proceedings of the Extended Abstracts on Human Factors in Computing Systems (CHI EA'14)*. ACM, New York, NY, 2527–2532. DOI: http://dx.doi.org/10.1145/2559206.2581181

Han Yu, Zhiqi Shen, and C. Leung. 2013. Bringing reputation-awareness into crowdsourcing. In *Proceedings of the 2013 9th International Conference on Information, Communications and Signal Processing (ICICS'13)*. 1–5. DOI: http://dx.doi.org/10.1109/ICICS.2013.6782912

Tao Yue, Shaukat Ali, and Shuai Wang. 2015. An evolutionary and automated virtual team making approach for crowdsourcing platforms. In *Crowdsourcing*, Wei Li, Michael N. Huhns, Wei-Tek Tsai, and Wenjun Wu (Eds.). Springer, Berlin, 113–130. DOI: http://dx.doi.org/10.1007/978-3-662-47011-4_7

Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. 2011. 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)*. IEEE Computer Society, Washington, DC, 409–412. DOI: http://dx.doi.org/10.1109/iThings/CPSCom.2011.128

Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. 2012a. Task recommendation in crowdsourcing systems. In *Proceedings of the 1st International Workshop on Crowdsourcing and Data Mining (CrowdKDD'12)*. ACM, New York, NY, 22–26. DOI: http://dx.doi.org/10.1145/2442657.2442661

Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. 2012b. TaskRec: Probabilistic matrix factorization in task recommendation in crowdsourcing systems. In *Proceedings of the 19th International Conference on Neural Information Processing, Volume II (ICONIP'12)*. Springer-Verlag, Berlin, 516–525. DOI: http://dx.doi.org/10.1007/978-3-642-34481-7_63

Omar F. Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (HLT'11)*. Association for Computational Linguistics, Stroudsburg, PA, 1220–1229.

Haoqi Zhang, Edith Law, Rob Miller, Krzysztof Gajos, David Parkes, and Eric Horvitz. 2012. Human computation tasks with global constraints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'12)*. ACM, New York, NY, 217–226. DOI: http://dx.doi.org/10.1145/2207676.2207708

Received December 2016; revised April 2017; accepted August 2017