**VLDB Journal manuscript No.**
(will be inserted by the editor)

# Task-Assignment Optimization in Knowledge Intensive Crowdsourcing

**Senjuti Basu Roy · Ioanna Lykourentzou · Saravanan Thirumuruganathan · Sihem Amer-Yahia · Gautam Das**

Received: date / Accepted: date

**Abstract** We present SmartCrowd, a framework for optimizing task-assignment in knowledge-intensive crowdsourcing (KI-C). SmartCrowd distinguishes itself by formulating, for the first time, the problem of worker-to-task assignment in KI-C as an optimization problem, by proposing efficient adaptive algorithms to solve it and by accounting for human factors, such as worker expertise, wage requirements, and availability inside the optimization process. We present rigorous theoretical analyses of the task-assignment optimization problem and propose optimal and approximation algorithms with guarantees, which rely on index pre-computation and adaptive maintenance. We perform extensive performance and quality experiments using real and synthetic data to demonstrate that the SmartCrowd approach is necessary to achieve efficient task assignments of high-quality under guaranteed cost budget.

**Keywords** collaborative crowdsourcing · optimization · knowledge-intensive crowdsourcing · human factors

The work of Saravanan Thirumuruganathan and Gautam Das is partially supported by NSF grants 0812601, 0915834, 1018865, a NHARP grant from the Texas Higher Education Coordinating Board, and grants from Microsoft Research and Nokia Research.

S. Basu Roy
University of Washington Tacoma
E-mail: senjutib@uw.edu

I. Lykourentzou
CRP Henri Tudor/INRIA Nancy Grand-Est
E-mail: ioanna.lykourentzou@{tudor.lu,inria.fr}

S. Thirumuruganathan
UT Arlington
E-mail: saravanan.thirumuruganathan@mavs.uta.edu

S. Amer-Yahia
CNRS, LIG
E-mail: sihem.amer-yahia@imag.fr

G. Das
UT Arlington
E-mail: gdas@uta.edu

## 1 Introduction

Knowledge-intensive crowdsourcing (KI-C) is acknowledged as one of the most promising areas of next-generation crowdsourcing [29], mostly for the critical role it can play in today's knowledge-savvy economy. KI-C refers to the collaborative creation of knowledge content (for example Wikipedia articles, or news articles) through crowdsourcing. Crowd workers, each having a certain degree of expertise, collaborate and "build" on each other's contributions to gradually increase the quality of each knowledge piece (hereby referred to as "task"). Despite its importance, no work or platform so far optimizes KI-C, a fact which often results in poor task quality and higher-than-expected costs, thus undermining the reliability of crowds for knowledge-intensive applications.

In this paper we propose SmartCrowd, an optimization framework for knowledge-intensive collaborative crowdsourcing that aims at improving KI-C by optimizing one of its fundamental processes, i.e., worker-to-task assignment, while taking into account the dynamic and uncertain nature of a real crowdsourcing environment [43].

Consider the example of a KI-C application offering news articles on demand as a service to interested stakeholders, such as publication houses, blogs, individuals, etc. Several thousands of workers are potentially

available to compose thousands of news articles collaboratively. It is easy to imagine that such an application needs to judiciously assign workers to tasks, so as to ensure the delivery of high-quality articles while being cost-effective. Two main challenges need to be investigated: 1) How to formalize the KI-C worker-to-task assignment problem? 2) How to solve the problem efficiently so as to warrant the desired quality/cost outcome of the KI-C platform, while taking into account the unpredictability of human behavior and the volatility of workers in a realistic crowdsourcing environment?

The paper is structured as follows: First, we formalize **KI-C worker-to-task assignment as an optimization problem** (Section 2). In our formulation, the resources are the worker profiles (knowledge skill per domain, requested wage) and the tasks are the news articles (each having a minimum quality and a maximum cost requirement, as well as the need for certain skills).[1] The objective function is formalized so as to guarantee that each task surpasses its quality threshold, stays below its cost limit, and that workers are not over or under utilized. Given the innate uncertainty induced by human involvement, we also use *probabilistic modeling* to include a third human factor, i.e. the workers' acceptance ratio[2], in the problem formulation.

Then, we argue that it may be prohibitively expensive to assign workers to tasks optimally in real time and reason about the necessity of *pre-computation* for efficiency reasons. We propose **index design as a means to efficiently address the KI-C optimization problem** (Section 3). One of the novel contributions of this work is proposing the pre-computation of *crowd indexes* (C-DEX) for KI-C tasks, to be used efficiently afterwards during the actual worker-to-task assignment process. We show how KI-C tasks could benefit from these pre-computed crowd indexes to efficiently maximize the objective function.

Third, we examine **the problem under dynamic conditions of the crowdsourcing environment**, where new workers may subscribe, existing ones may leave, worker profiles may change over time, and workers may accept or decline recommended tasks. To tackle such unforeseen scenarios, SMARTCROWD proposes the *adaptive maintenance of the pre-computed indexes*, while respecting worker non-preemption.[3]

[1] With the availability of historical information, worker profiles (knowledge skills and expected wage) can be learned by the platform. Profile learning is an independent research problem in its own merit, orthogonal to this work.

[2] Acceptance ratio of a worker is the probability that she accepts a recommended task.

[3] Non-preemption ensures that a worker cannot be interrupted after she is assigned to a task.

Fourth, we prove **several theoretical properties of the** C-DEX **design problem**, such as *NP-Completeness* (using a reduction from the Multiple Knapsack Problem [13]), as well as *submodularity* and *monotonicity* under certain conditions. This in-depth theoretical analysis is critical to understand the problem complexity, as well as to design efficient principled solutions with theoretical guarantees.

Finally, we propose **novel optimal and approximate solutions for the index design and maintenance, depending on the exact KI-C problem conditions.** Our optimal solution uses an integer linear programming (ILP) approach (Section 4). For the case where the optimal index building or maintenance is too expensive, we propose *two types of efficient approximate strategies*: 1) a greedy approach (Section 5) consisting of one randomized and one deterministic approximation algorithm, which both need polynomial computation time and admit constant approximation factors under certain conditions (2/5 for the randomized algorithm and $(1 - 1/e)$ for the deterministic one) and 2) a clustering-based approximation approach (Section 6), called C-DEX$^+$, which is based on the idea of building and maintaining the indexes using clusters of similar workers (a notion called "virtual worker") instead of the actual worker pool.

We design comprehensive experimental studies (Section 7), both with real-users and simulations, to validate SMARTCROWD qualitatively and efficiency-wise. With an appropriate adaptation of Amazon Mechanical Turk (AMT), we conduct extensive quality experiments involving real workers to compose news articles. Such an adaptation needs a careful design of the validation strategies, since AMT (like many other popular paid crowdsourcing platforms) does not yet support KI-C tasks. Extensive simulation studies are used to further investigate quality and efficiency. In these, we compare against several baseline algorithms, including one of the latest state-of-the-art techniques [18] for online task assignment in crowdsourcing. The obtained results demonstrate that *the algorithms proposed by the* SMARTCROWD *framework achieve 3x improvement*, both qualitatively and efficiency-wise, justifying the necessity for pre-computed indexes and their adaptive maintenance for the KI-C optimization problem.

Our main contributions are summarized as follows:

1. We initiate the study into task assignment optimization in knowledge-intensive crowdsourcing (KI-C), formalize the problem, and propose rigorous theoretical analyses.
2. We propose the necessity of index design and dynamic maintenance to address the KI-C task assignment optimization problem. We propose novel

optimal and approximate solutions (C-DEX, greedy C-DEX, and C-DEX$^+$) for index creation and adaptive maintenance.
3. We conduct extensive experiments on real and simulated crowdsourcing settings to demonstrate the effectiveness of our proposed solution qualitatively and efficiency-wise.

The rest of this paper is organized as follows. Section 2 describes the KI-C task assignment optimization problem, including a description of its data model, its constraints and its objective. Section 3 presents the SMARTCROWD framework, which maps the KI-C task assignment problem to an index design problem (C-DEX design problem) and analyzes the latter theoretically. Section 4 describes the proposed optimal algorithm (C-DEX-Optimal). Section 5 describes the two greedy approximation algorithms (C-DEX-Randomized and C-DEX-Deterministic) and Section 6 presents the clustering-based approximation strategy (C-DEX$^+$). Section 7 contains the experiments, Section 8 presents the related work and Section 9 triggers a discussion on the extensions and future perspectives. Finally Section 10 concludes the paper.

## 2 KI-C Problem Settings

In this section, we describe a framework to formalize knowledge intensive crowdsourcing (KI-C). KI-C is widely considered as a key component of next generation of crowdsourcing. We refer the reader to Section 8 for its description and a brief survey of KI-C. For most of our paper, we use the application of collaborative document editing to illustrate our model and algorithms. We detail how our framework could be utilized for other popular KI-C applications such as fansubbing (sentence translation by fans) in Section 9.4.

### 2.1 Data Model

We are given a set of workers $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$, a set of skills $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ and a set of tasks $T = \{t_1, t_2, \ldots, t_l\}$. In the context of collaborative document writing, skills represent topics such as Egyptian Politics, PlayStation games, or the NSA document leakage. Tasks represent the documents that are being edited collaboratively.

*Skills:* A skill is the knowledge on a particular topic, quantified in a continuous scale between $[0, 1]$. It is associated to workers and tasks. When associated to a worker, it represents the worker's expertise of a topic. When associated to a task, a skill represents the minimum quality requirement for that task. A value of 0 for a skill reflects no expertise of a worker for that skill. For a task, 0 reflects no requirement for that skill.

| Notation | Interpretation |
|---|---|
| $\mathcal{U}$ | the set of $n$ workers |
| $\mathcal{S}$ | the set of $m$ skills |
| $T$ | the set of $l$ tasks |
| $t = \langle Q_{t_1}, Q_{t_2}, \ldots, Q_{t_m}, W_t \rangle$ | a task vector with quality and cost thresholds |
| $u = \langle u_{s_1}, u_{s_2}, \ldots, u_{s_m}, w_u, p_u \rangle$ | profile of worker $u$, $m$ skills, wage, and acceptance ratio |
| $v_j$ | value of task $j$ |
| $i^t = \langle \mathcal{P}_i, \mathcal{L}_i \rangle$ | a C-DEX for task $t$ |
| $\mathcal{I}$, $\mathcal{I}_v$ | a set of C-DEX, C-DEX$^+$ |
| $\mathcal{C}_u$ | the number of active tasks assigned to worker $u$ |
| $C_1, C_2$ | two constant weights associated to skill and cost respectively |
| $X_l, X_h$ | minimum and maximum (respectively) task load of a worker |

Table 1: Major notations used in the paper

*Workers:* Each worker $u \in \mathcal{U}$ has a *profile* that is a vector, $\langle u_{s_1}, u_{s_2}, \ldots, u_{s_m}, w_u, p_u \rangle$, of length $m + 2$ describing her $m$ skills in $\mathcal{S}$, her wage $w_u$, and her task acceptance ratio $p_u$.

- Skill $u_{s_i} \in [0, 1]$ is the expertise level of worker $u$ for skill $s_i$. Skill expertise reflects the quality that the worker's contribution will assign to a task accomplished by that worker.
- Wage $w_u \in [0, 1]$ is the minimum amount of money for which a worker $u$ is willing to accept to complete a given task.
- Acceptance ratio $p_u \in [0, 1]$ is the probability at which a worker $u$ accepts a task. It reflects the worker's availability to complete tasks assigned to her. A value of 0 is used to model workers who are not available (as workers who do not accept any task).

We refer to a worker's skill, wage expectation and acceptance ratio as *human factors* that may vary over the time.

*Tasks:* A task $t \in T$ is characterized by a vector, $\langle Q_{t_1}, Q_{t_2}, \ldots, Q_{t_m}, W_t \rangle$ of length $m+1$, which reflects the task's minimum quality requirement per skill and its maximum cost (or maximum allowed wage). A task $t$ that is being executed has a set of contributors $\mathcal{U}_t \subseteq \mathcal{U}$ so far and it is characterized by a:

- Current quality $q_{t_i} = \Sigma_{u \in \mathcal{U}_t} u_{s_i} \in [0, |\mathcal{U}_t|]$ for skill $s_i$, with $u_{s_i}$ being the expertise of worker $u$ on skill $s_i$. The current quality $q_{t_i}$ is thus the aggregate of the skill $i$ of all workers who have contributed to the task $t$ so far.
- Current cost $w_t = \Sigma_{u \in \mathcal{U}_t} w_u \in [0, |\mathcal{U}_t|]$, with $w_u$ being the wage paid to worker $u$. $w_t$ aggregates the wages of all workers who have contributed to $t$ so far.

Notice that in our paper, we have defined task quality as the *sum* of workers skills who take part in it. This is commonly known as the *additive* skill aggregation model [2] that is commonly used in KI-C tasks such as document editing, fan-subbing (sentence translation by fans). In additive skill aggregation model, the quality of final task is proportional to the sum of worker skills. This simple intuitive function for transforming individual contributions into a collective result has been adopted in many previous KI-C tasks[2,34], where workers' build on each others' contribution by performing edits. Of course, other aggregation measures such as maximum, minimum, or product [2] could also be used to compute the quality of a task. However, their use in KI-C tasks is less obvious.

We would also like to note that the acceptance ratio of a worker has an impact on how the current quality and cost of a task is computed. If all the workers assigned to a task are available, the current quality and cost of the task is simply the sum of worker skills and wages respectively. However, it is possible that when a team is formed, some of the workers might not be available. For example, a worker $u$ with acceptance ratio of 0.5 could only be available 50% of the time. We can extend the definition to handle this uncertainty in a straightforward manner. We now compute the *expected* quality and cost of a task which is computed by a *weighted* sum of worker skills and wages with their respective acceptance ratio acting as the weight.

*Workload:* We assume a static workload $T$ that represents a set of active tasks over a given time period. Each task in the workload is associated with both minimum quality requirement per skill and a maximum cost. In the spirit of database workloads, we assume that a crowdsourcing workload is representative of the type of tasks handled by the crowdsourcing platform. Existing techniques for capturing database workloads are equally applicable for estimating crowdsourcing workloads.

## 2.2 Constraints

The following constraints are considered:

Table 2: Workers Profiles

| **Worker** | $\mathbf{u_1}$ | $\mathbf{u_2}$ | $\mathbf{u_3}$ | $\mathbf{u_4}$ | $\mathbf{u_5}$ | $\mathbf{u_6}$ |
|---|---|---|---|---|---|---|
| **Skill** | 0.1 | 0.3 | 0.2 | 0.6 | 0.4 | 0.5 |
| **Wage** | 0.05 | 0.25 | 0.3 | 0.7 | 0.3 | 0.4 |
| **Acceptance ratio** | 0.8 | 0.7 | 0.8 | 0.5 | 0.6 | 0.9 |

Table 3: Task Descriptions

| **Task** | $\mathbf{t_1}$ | $\mathbf{t_2}$ | $\mathbf{t_3}$ |
|---|---|---|---|
| **Quality threshold** | 0.7 | 0.7 | 0.9 |
| **Cost threshold** | 1.08 | 1.1 | 2.0 |

- **Minimum Quality:** For each task $t \in T$, the worker-to-task assignment has to be such that the aggregated skill of assigned workers is at least as high as the minimum skill requirement of $t$ for each skill. [4]
- **Maximum Cost:** For each task $t \in T$, the aggregated workers' wage ($w_t$) cannot exceed the maximum cost that $t$ can pay, i.e., $w_t \leq W_t$.
- **Non-preemption.** Once a worker has been assigned to a task, she cannot be pulled out of that task until finished.
- **Minimum/Maximum Tasks per worker:** A worker must be assigned a minimum number of $X_l$ tasks and no more than $X_h$ tasks.

## 2.3 Objective

Given a set $T$ of tasks and a set $\mathcal{U}$ of workers, the objective is to perform worker-to-task assignment for all tasks in $T$, such that the overall task quality is maximized and the cost is minimized, while the constraints of skill, cost, and tasks-per-worker are satisfied.

*Example 1* We describe a running example consisting of a minuscule version of the news article composition task. Assume that the platform consists of 6 workers to compose 3 news articles (tasks) on "Egyptian Politics"($t_1$), "NSA leakage"($t_2$) and "US Health Care Law" ($t_3$). For simplicity, we assume that all tasks belong to same topic ("Politics") and therefore require only one skill (knowledge in politics). We also assume that $X_l = 1$, $X_h = 2$. Worker profiles (skill, wage,acceptance ratio) and task requirements (minimum quality, maximum cost) are depicted numerically in Tables 2 and 3. This example will be used throughout the paper to illustrate our solution.

[4] $Q_{t_j}$ is the threshold for skill $j$ and $q_{t_j} \geq Q_{t_j}$.

## 3 SmartCrowd

The SmartCrowd framework maps the KI-C optimization problem to a problem of index building and maintenance. In the following we formalize the index design problem and analyze it in-depth theoretically. Then we highlight the need for adaptive index maintenance. Finally we close this section by presenting the unified approach that all SmartCrowd algorithms adopt towards solving the problem.

### 3.1 C-DEX

**C-DEX Design Problem:** We now formally describe the C-Dex design problem. C-Dex stands for Crowd-Index which is, intuitively, a pre-computation of task assignments for a given worker pool and workload. The pre-computed assignments facilitate faster worker to task assignment (akin to indexing in Databases). Given a new task, C-dex could be used to make a fast "lookup" to find a team of workers best suited for the task.

We start with the KI-C objective described in Section 2.3. We define $v_t$ to denote the *value* of each task $t$ in $T$ (in the beginning $v_t$ is 0 for every task). The task value is associated with the current quality and cost of the task. More specifically, task value is calculated as a weighted linear combination of skills (higher is better) and cost (lower is better). Also recall that each task has a minimum skill requirement and a maximum cost budget. The index associated with task $t$ should be created such that the aggregated skill of the assigned workers is at least as large as the minimum skill requirement of $t$ for each skill [5], and that the aggregated workers' wage does not exceed the maximum cost that $t$ can pay (i.e., $w_t \leq W_t$). Once these two hard-constraints are satisfied, a positive *value* $v_t$ is associated to task $t$. The objective is to design an index C-dex such that the sum of values $\mathcal{V} = \Sigma_{t \in T} v_t$ of all tasks $T$ is maximized, while the problem constraints, as set in Section 2.2 are satisfied.

For a task $t$, its individual value $v_t$ and the global value $\mathcal{V}$ of the objective function are defined in Equation 1.

$$\text{Maximize } \mathcal{V} = \sum_{t \in T} v_t \qquad \text{subject to:} \tag{1}$$

$$\forall t \in T \quad v_t = \begin{cases} C_1 \times \sum_{1 \leq j \leq m} q_{t_j} + C_2 \times (1 - \frac{w_t}{W_t}) \\ \text{if } q_{t_j} \geq Q_{t_j} \text{ and } w_t \leq W_t \\ 0 \qquad \text{otherwise} \end{cases}$$

where $C_1, C_2 \geq 0$ and $C_1 + C_2 = 1$.
The above formulation is a flexible incorporation of different skills and cost, letting the application select the respective weights $(C_1, C_2)$, as appropriate.

Since C-dex are *pre-computed for future use*, the skills (or quality) and wages are computed in an expected sense considering the workers' acceptance ratio, instead of actual aggregates, as follows:

$$\forall t \in T, \forall 1 \leq j \leq m \qquad q_{t_j} = \sum_{u \in \mathcal{U}} u_t \times p_u \times u_{s_j}$$

$$\forall t \in T, \qquad w_t = \sum_{u \in \mathcal{U}} u_t \times p_u \times w_u$$

$$\forall t \in T, \forall u \in \mathcal{U}, \qquad u_t \in \{0, 1\}$$

$$\forall u \in \mathcal{U}, \qquad X_l \leq \sum_{t \in T} u_t \leq X_h$$

**Solution Format** The solution to the above problem formulation is a set $\mathcal{I}$ of C-dex indexes, one index for each task $t \in \mathcal{T}$. Each C-dex index $i^t$ contains the estimated properties $\mathcal{P}_i^t$ of its respective task $t$ (which include the task's value $v_t$, estimated quality per skill $q_{t_1}, \ldots, q_{t_m}$, and final estimated cost $w_t$) and the set $\mathcal{L}_i^t$ of users who are assigned to the task.

We define the crowd index C-dex as follows:

**Definition 1** (C-dex) A C-dex $i^t = (\mathcal{P}_i^t, \mathcal{L}_i^t)$ is a pair that represents an assignment of a set of workers in $\mathcal{U}$ to a task $t$. Formally, it is described by a vector $\mathcal{P}_i^t$ of length $m + 2$, and a set of workers $\mathcal{L}_i^t$. $\mathcal{P}_i^t = \langle v_t, q_{t_1}, \ldots, q_{t_m}, w_t \rangle$ contains the value $v_t$ of task $t$, its expected total expertise $q_{t_i}$ for each skill $s_i$, and its expected total cost $w_t$. $\mathcal{L}_i^t \subseteq \mathcal{U}$ contains the workers assigned to index $i^t$.

Consider Example 1 with $T = \{t_1, t_2, t_3\}$ for which three indexes are to be created offline. If workers $\{u_1, u_2, u_6\}$ are assigned to task $t_1$ with $C_1 = C_2 = 0.5$, then the index for task $t_1$ will be, $i^{t_1} = (\langle 0.6, 0.74, 0.58 \rangle, \{u_1, u_2, u_6\})$. The expected quality of $t_1$ is computed by multiplying the skills of assigned workers with their acceptance ratio. For this assignment, the expected quality is $0.1 \times 0.8 + 0.3 \times 0.7 + 0.5 \times 0.9 = 0.74$. Similarly, the expected cost is computed as $0.05 \times 0.8 + 0.25 \times 0.7 + 0.4 \times 0.9 \approx 0.58$. Finally, the value of this particularly assignment is $0.5 \times 0.74 + 0.5 \times (1 - 0.58/1.08) = 0.6$.

[5] $Q_{t_j}$ is the threshold for skill $j$ and $q_{t_j} \geq Q_{t_j}$.

*3.1.1 Theoretical Analyses*

**Theorem 1** *The decision version of* C-DEX *Design Problem is NP-Complete.*

*Proof* Given a workload $T$ of $l$ tasks, a set of workers (and their profiles), constant values $C_1, C_2, X_l, X_h$ and $\mathcal{V}$, the decision version of the problem seeks if a set of $l$ indexes could be created (one for each task), where $v_t$ is the value of index $i^t$, such that all constraints are satisfied and the aggregated global value $\sum_{i=1}^{l} v_t^i$ greater than or equal to $\mathcal{V}$.

It is easy to see that the problem is in NP. To prove NP-completeness, we prove that the well known Partition [13] problem is polynomial time reducible to an instance of the C-DEX Design Problem, i.e., Partition $\leq_p$ C-DEX Design Problem.

The decision version of the Partition problem is as follows: given a finite multiset $\mathcal{A}$ of positive integers, can $\mathcal{A}$ be partitioned into two disjoint subsets $\mathcal{A}_1$ and $\mathcal{A}_2$ such that the sum of the numbers in $\mathcal{A}_1$ (i.e., $S(\mathcal{A}_1)$) equals the sum of the numbers in $\mathcal{A}_2$ (i.e., $S(\mathcal{A}_2)$).

We reduce an instance of Partition to create an instance of the C-DEX Design Problem, as follows. Each number in the multiset represents the skill of an individual worker $u$ (number of skill domain $m = 1$) and is scaled down to a rational number between $[0-1]$ by diving it by the maximum integer ($I_{max}$) in the multiset. We assign the wage of each $u$ to be 0, and acceptance ratio to be 1, i.e., $p_u = 1$. The workload consists of 2 tasks (is equal to the number of indexes). Both tasks have a minimum and equal skill requirement, i.e., for task $t$, $Q_t = S(\mathcal{A}_1)/I_{max} = S(\mathcal{A}_2)/I_{max} = S(\mathcal{A})/(2 * I_{max})$. Since each worker does not have a cost associated, the cost constraint $W_t$ for task $t$ can be set arbitrarily. Each partition represents an C-DEX and the aggregated skill of the workers inside the index must be equal or more than $S(\mathcal{A}_1)/I_{max}$ or $S(\mathcal{A}_2)/I_{max}$. The constant weights $C_1$ and $C_2$ are chosen arbitrarily, as long as $C_1+C_2 = 1$.

This creates the following instance of the C-DEX Design problem, where $v_j$ is the value of the $j$-th C-DEX, and $\mathcal{V}$ is the overall value:

$$\mathcal{V} = v_1 + v_2$$
$$v_j = C_1 \times q_j + C_2 \times (1 - \frac{0}{W_j}),$$
$$q_j = \sum_{u \in \mathcal{U}} u_j \times p_u \times u_s,$$
$$w_j = 0, X_l = X_h = 1,$$
$$Q_j = S(\mathcal{A}_1)/I_{max} = S(\mathcal{A}_2)/I_{max},$$
$$q_j \geq Q_j, \forall j \in \{1, 2\}$$

Given the above instance of the C-DEX Design Problem, the objective is to create 2 C-DEX, such that $\mathcal{V} = C_1 \times \frac{S(\mathcal{A}_1)+S(\mathcal{A}_2)}{I_{max}} + C_2$ and there exists a solution of the Partition problem, if and only if, a solution to our instance of the C-DEX Design Problem exists.

*3.1.2 Effects of the Constraints on the C-DEX Design Problem*

We investigate interesting theoretical properties of the optimization problem (Equation 1) under different conditions and constraints. In particular, we investigate the submodularity and monotonicity properties [38] of the objective function, which are heavily used to design approximation algorithms with theoretical guarantees in Section 5. Specifically, next we prove that our value function $v_t$ and our objective function $\mathcal{V}$ are neither submodular nor monotonic in the general case. These results prevent the design of approximation algorithms with theoretical guarantees for our problem in the general case. However, under special conditions of the constraints the value function and the objective function become submodular. Finally, as we show next, monotonicity could be ensured for these functions, when the weight value $C_2$ on cost becomes 0.

*Submodular Function:* In general, if $\mathcal{A}$ is a set, a submodular function is a set function: $f : 2^{\mathcal{A}} \rightarrow \mathbb{R}$ that satisfies the following condition: For every $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{A}$ with $\mathcal{X} \subseteq \mathcal{Y}$ and every $x \in \mathcal{A} \backslash \mathcal{Y}$, we have $f(\mathcal{X} \cup \{x\}) - f(\mathcal{X}) \geq f(\mathcal{Y} \cup \{x\}) - f(\mathcal{Y})$.
The value function $v_t$ for task $t$ satisfies this form: it maps each subset of the workers $\mathcal{S}$ from $\mathcal{U}$ to a real number $v_t$, denoting the value that we get if that subset of workers are assigned to task $t$. Conversely, the global optimization function $\mathcal{V} = \Sigma_{t \in T} v_t$ is defined over a set of sets, where each set maps an assignment of a subset of the workers from $\mathcal{U}$ to a task $t \in T$ with value $v_t$.

*Monotonic Function:* A function $f$ defined on non-empty subsets is monotonic if for every $\mathcal{X} \subseteq \mathcal{Y}$, $f(\mathcal{X}) \leq f(\mathcal{Y})$.

**Theorem 2** *The value function $v_t$ is not submodular in the* C-DEX *Design problem, if $Q_{t_j} > 0, \forall j \in \{1..m\}$.*

*Proof* Sketch: Without loss of generality, we ignore the weights and the acceptance ratios of the workers for this proof. For the simplicity of exposition imagine $m = 1$. The value $v_t$ is defined both on quality and cost, but remains 0, if $q_{t_1} < Q_{t_1}$, or $w_t > W_t$, or both. Therefore, for the rest of our argument, we assume an infinite cost budget and only focus on quality. Under this assumption, $v_t$ remains 0, until $q_{t_1}$ becomes $\geq Q_{t_1}$.

Consider a subset $\mathcal{R} \subset \mathcal{S}$ and imagine $f'(\mathcal{R}) < Q_{t_1}$, leading to $v_t = 0$. If an element $k$ is added to $\mathcal{R}$, if

$f'(\mathcal{R} \cup k) < Q_{t_1}$, then $v_t = 0$. However, if $f'(\mathcal{S}) \geq Q_{t_1}$, $v_t > 0$. Therefore, $f'(\mathcal{S} \cup k) - f'(\mathcal{S}) > 0$. In such cases, it is easy to see,
$f'(\mathcal{S} \cup k) - f'(\mathcal{S}) > f'(\mathcal{R} \cup k) - f'(\mathcal{R})$. This clearly violates the submodularity condition. We omit the details for brevity.

**Theorem 3** *The value function $v_t$ in the* C-DEX *Design problem is submodular but non-monotone, when $Q_{t_j} = 0, \forall j \in \{1..m\}$.*

*Proof* Sketch: As long as $Q_{t_j} = 0, \forall j \in \{1..m\}$ (meaning no quality threshold is provided), it could be proved that the increase in value by adding a worker $k$ to $\mathcal{S}$ is less or equal to adding $k$ to $\mathcal{R}$, where $\mathcal{R} \subset \mathcal{S}$, with the cost threshold $w_t \leq W_t$. Therefore, the following condition of submodularity i.e., "diminishing return" holds: $f(\mathcal{S} \cup k) - f(\mathcal{S}) \leq f(\mathcal{R} \cup k) - f(\mathcal{R})$. At the same time, $v_t$ could increase or decrease when a worker is added (depending on whether the skill increase is more than the cost decrease or vice versa). Hence $v_t$ is non-monotone.

Consider our running example (Example 1) and note that the value function for task $t_1$ will be submodular but non-monotone when quality threshold is changed to 0, i.e., $Q_{t_1} = 0$, instead of 0.7.

**Theorem 4** *The value function $v_t$ and the objective function $\mathcal{V}$ in the* C-DEX *Design problem are submodular and monotonic, when $Q_{t_j} = 0, \forall j \in \{1..m\}$ and $C_2 = 0$.*

*Proof* Sketch: Consider Theorem 3 that proves the submodularity property of $v_t$ when $Q_{t_j} = 0$. It is easy to see that when $C_2 = 0$, $v_t$ will only strictly increase with the addition of workers. This ensures the monotonicity of $v_t$.

Next, consider our objective function $\mathcal{V} = \Sigma_{t \in T} v_t$ defined over a set of sets, where each set defines a subset of workers assigned to a task $t$ with value $v_t$. Adding a worker $k$ to a set $\mathcal{R}$ (corresponds to task $t$) will impact $v_t$ and therefore the overall $\mathcal{V}$. Without the skill threshold, i.e., $Q_{t_j} = 0, \forall j \in \{1..m\}$, if $k$ is added to $\mathcal{S}$ instead, where $\mathcal{R} \subset \mathcal{S}$, the following condition of submodularity will hold: $f(\mathcal{S} \cup k) - f(\mathcal{S}) < f(\mathcal{R} \cup k) - f(\mathcal{R})$. Furthermore, $\mathcal{V}$ strictly increases when $C_2 = 0$ and ensures monotonicity.

Consider our running example (Example 1) again, and note that the value function $v_{t_1}$ for task $t_1$ will be both submodular and monotone when the quality threshold is changed to 0, and the cost function in the objective function (i.e., $C_2 \times (1 - \frac{w_t}{W_t})$) becomes 0 by setting $C_2 = 0$. This means that the objective function only wishes to maximize the skill while satisfying only the cost threshold. Similarly, the global objective function $\mathcal{V}$ will become submodular as well as monotone when all three tasks have quality threshold as 0 and have $C_2 = 0$.

### 3.2 Index Maintenance

Indexing workers in KI-C is more challenging than data indexing for query processing, due to the human factors involed in a dynamic crowdsourcing environment. In particular, a unique challenge that SMARTCROWD faces is that even if the most appropriate index is selected for a task, one or more workers who were assigned to the task may not be available (for example, they are not online or they decline the task). The acceptance ratio only quantifies an overall availability of a worker, but not for a particular task. Therefore, SMARTCROWD needs to dynamically find a replacement for unavailable workers. At the same time, SMARTCROWD needs to strictly ensure non-preemption of the workers, since it is not desirable that the workers who have already accepted a task and are currently working on it to be forced to stop their current assignment in order to be reassigned to different tasks.

Furthermore, SMARTCROWD has to deal with scenarios where new workers may subscribe to the system any time or some existing ones may delete their accounts. Similarly, as existing workers complete more tasks, the system may update their profile (refine their skills for example). How to learn the profile of a new worker or an updated profile of an existing worker is orthogonal to this work. What we are interested in here is *how* SMARTCROWD *makes use of these dynamic updates, by maintaining the indices incrementally.*

We will therefore need to investigate a principled solution towards incremental index maintenance for four scenarios: (1) worker replacement due to unavailability for the task, (2) worker addition, (3) worker deletion, (4) worker profile update.

### 3.3 Sketching the solution

Taking into the account the above-presented problem characteristics, as well as it theoretical analysis, we now proceed with sketching the solution.

SMARTCROWD adopts a unified approach to solve the C-DEX design and maintenance problem and the overall functionality of its algorithms is as follows:

- **1. Offline Phase - Index Building:** A set of indexes $\mathcal{I}$, referred to as C-DEX are pre-computed based on a simple definition of past task *workload.* This step is referred to as the offline phase.

– **2. Online Phase - Index Use and Maintenance:** The pre-computed indexes are used to perform efficient worker-to-task assignments once the actual tasks arrive. The pre-computed indexes are also *maintained adaptively* to account for worker replacements, additions, deletions or profile updates, while respecting worker non-preemption. This step is referred to as the online phase.

Of course, if the actual tasks are substantially different from the workload, SmartCrowd has to halt and redesign the indexes from scratch. The latter scenario is orthogonal to us.

Taking into account the above, in the next sections we proceed as follows. First we propose an optimal (i.e., exact) solution in section 4. Next, in section 5, we propose two approximate algorithms, each of which uses a greedy C-dex building and maintenance strategy and admits a provable approximation factor under certain conditions. Then, in section 6 we propose an alternative index building and maintenance algorithm, namely C-dex$^+$, which is based on clustering.

## 4 Optimal Algorithm

We describe the optimal (i.e., exact) C-dex building solution in Section 4.1 and we discuss it maintenance in Section 4.2.

### 4.1 C-DEX-Optimal Design (offline phase)

Recall Theorem 1 and note that the C-dex Design Problem is proved to be NP-hard. SmartCrowd proposes an integer linear programming (ILP)-based solution that solves the optimization problem defined in Equation 1 optimally satisfying the constraints. Our implementation uses the primal-dual barrier method [45] to solve the ILP.

While the optimization problem is a linear combination of weights and skills, unfortunately, the decision variables (i.e., $u_t$'s) are required to be integers. More specifically, C-dex sets are created by generating a total of $n \times |T|$ boolean decision variables, and the solution of this optimization problem assigns a 1/0 value to each variable, denoting that a worker is assigned to a particular task, or not. These integrality constraints make the above formulation an Integer Linear Programming (ILP) problem [15]. A solution to the ILP problem performs an assignment of a worker to a task in $T$. Once the optimization problem is solved, an index $i^t$ is designed for each task in the workload and $\langle \mathcal{P}_i^t, \mathcal{L}_i^t \rangle$ is calculated. Algorithm 1 summarizes the pseudocode.

**Algorithm 1** Optimal C-dex Design Algorithm

**Input:** Workload $T$
1: Solve the C-dex Design ILP to get an assignment of the $u_t \in \{0, 1\}$, where $u$ is a worker, and $t \in T$.
2: using $u_t$, for each $t \in T$, compute and output $i^t = \langle \mathcal{P}_i^t, \mathcal{L}_i^t \rangle$
3: **return** Index set $\mathcal{I}$

Given Example 1, when $C_1 = C_2 = 0.5$, the best allocation gives rise to $\mathcal{V} = 1.94$, with the following worker to task allocation: $u_1 = \{t_1\}, u_2 = \{t_1, t_2\}, u_3 = \{t_3\}, u_4 = \{t_2, t_3\}, u_5 = \{t_2, t_3\}, u_6 = \{t_1, t_3\}$. This creates the following 3 indexes:
$i^{t_1} = (\langle 0.6, 0.74, 0.58 \rangle, \{u_1, u_2, u_6\})$,
$i^{t_2} = (\langle 0.55, 0.75, 0.71 \rangle, \{u_2, u_4, u_5\})$,
and $i^{t_3} = (\langle 0.79, 1.15, 1.13 \rangle \{u_3, u_4, u_5, u_6\})$.

Unfortunately, ILP is also NP-Complete [13]. The commercial implementations of ILP use techniques such as Branch and Bound [15] with the objective to speed up the computations. Yet, computation time is mostly non-linear to the number of associated variables and could become exponential in the worst case.

### 4.2 C-DEX-Optimal Maintenance (online phase)

We design index maintenance algorithms, which generate optimal solutions under the non-preemption constraint (constraint no.3, Section 2.2). Non-preemption of workers enforces that the existing assignment of an available worker can not be disrupted, only new assignments can be made if the worker is not maxed-out. Under this assumption, all four incremental maintenance strategies described below are optimal.

#### *4.2.1 Replacing Workers*

To dynamically find a replacement for unavailable workers, without disrupting already made assignments, we formulate a *marginal ILP* and solve the problem optimally only with the available set of workers.

We illustrate the scenario with an example. Suppose that after the most appropriate index $i^t$ is selected for task $t = \langle Q_{t_1}, Q_{t_2}, \ldots, Q_{t_m}, W_t \rangle$ using Equation 1, a subset of workers in $\mathcal{L}_i^t$ is unavailable or declines to work on $t$. Imagine that the quality of $i^t$ declines to $q'_{t_j}$ from $q_{t_j}$, for skill $j$, $\forall j \in [1, m]$, and the cost declines to $w'_t$ from $w_t$, since some workers do not accept the task. Consequently, the value of $i^t$ also declines, let us say, to $v'_t$ from $v_t$. From the worker pool $\mathcal{U}$, let us imagine that a subset of workers $\mathcal{U}'$ are available and their current assignment has not maxed out (i.e., $\mathcal{C}_{u'} < X_h$). To find the replacement of the unavailable workers, SmartCrowd works as follows: It formulates

a marginal ILP problem with the same optimization objective for $t$, only with the workers in $\mathcal{U}'$. More formally, the task is formulated as:

$$\text{Maximize } v''_t = v'_t + C_1 \times \sum_{\forall 1 \le j \le m} q''_{t_j} + C_2 \times (1 - \frac{w''_t}{W_t}) \quad (2)$$

$$q''_{t_j} = q'_{t_j} + \Sigma_{u' \in \mathcal{U}'} u'_j \times p_{u'} \times u_{s_j}$$
$$w''_t = w'_t + \Sigma_{u' \in \mathcal{U}'} u'_t \times p_{u'} \times w_{u'}, u'_t \in \{0,1\}.$$

**Lemma 1** *The marginal ILP in Equation 2 involves only $|\mathcal{U}'|$ variables.*

The above optimization problem is formulated only for a task $t$ and considering only $|\mathcal{U}'| \ll |\mathcal{U}|$ workers. It is incremental in nature, as it "builds" on the current solution (notice that it uses the declined cost, skills, and value in the formulation), involving a much smaller number of variables and leading to small latency. Moreover, this strategy is fully aligned with the optimization objective that SmartCrowd proposes. After this formulation is solved, $\mathcal{L}_i^t$ is updated with the new workers for which the above formulation has produced $u'_t = 1$.

#### *4.2.2 Adding New Workers*

Assume that a set $\mathcal{A}$ of new workers has subscribed to the platform. The task for SmartCrowd is to decide whether (or not) to assign those workers to any task in $T$, and if yes, what should be the assignment. Note that SmartCrowd already has assigned the existing worker set $\mathcal{U}$ to the tasks in $T$ and they can not be preempted.

The overall idea is to solve optimally a marginal ILP only with the new workers in $\mathcal{A}$ and tasks $T$, without making any modifications to the existing assignments of the $\mathcal{U}$ workers to the $T$ tasks. Formally, the problem is formulated as follows:

$$\text{Maximize } \Sigma_{t \in T} v'_t \quad (3)$$

$$v'_t = v_t + C_1 \times \sum_{\forall 1 \le j \le m} q'_{t_i} + C_2 \times (1 - \frac{w'_t}{W_t})$$
$$q'_{t_j} = \{q_{t_j} + \Sigma_{u \in \mathcal{A}} u_t \times p_u \times u_{s_j}\}$$
$$w'_t = \{w_t + \Sigma_{u \in \mathcal{A}} u_t \times p_u \times w_u\}$$
$$u_t \in \{0,1\}, 0 \le \Sigma_{t \in T}\{u_t \in \mathcal{A}\} \le X_h.$$

**Lemma 2** *The optimization problem in Equation 3 involves only $|\mathcal{A}| \times |T|$ variables.*

#### *4.2.3 Deleting Workers*

In principle, the treatment of worker deletion is analogous to that of worker replacement strategies in Section 4.2.1. Basically, the idea is to determine the decreased quality, cost, and value of each of the tasks that are impacted by the deletion, and then re-formulate an optimization problem only with those tasks, and the remaining workers who are not maxed-out yet (i.e., $\mathcal{C}_u < X_h$) on their assignment, using the current quality, cost, and value. Similar to Section 4.2.1, this formulation is also a marginal ILP that is incremental in nature, and involves a smaller number of variables. We omit further discussion on this for brevity.

#### *4.2.4 Updating Worker Profiles*

Interestingly, the handling of worker profile updates is also incremental in SmartCrowd. If the skill, wage, or acceptance-ratio of a subset $\mathcal{A}'$ of workers gets updated, SmartCrowd first updates the respective *value* of the tasks (where these workers were assigned), by discounting the contribution of the workers in $\mathcal{A}'$. After that, a smaller optimization problem is formulated involving only $\mathcal{A}'$ workers and $T$ tasks. After discounting the contribution of the workers in $\mathcal{A}'$, if the latest value of a task $t$ is $v'_t$[6], current quality on skill $j$ is $q'_{t_j}$, and current cost is $w'_t$, then the optimization problem is formulated as follows:

$$\text{Maximize } \Sigma_{t \in T}\{v'_t\} \quad (4)$$

where,

$$v''_t = v'_t + C_1 \times \sum_{\forall 1 \le j \le m} q''_{t_j} + C_2 \times (1 - \frac{w''_t}{W_t}),$$
$$q''_{t_j} = \{q'_{t_j} + \Sigma_{u \in \mathcal{A}'} u_t \times p_u \times u_{s_j}\}$$
$$w''_t = \{w'_t + \Sigma_{u \in \mathcal{A}'} u_t \times p_u \times w_u\}$$
$$u_t = \{0,1\}, X_l \le \Sigma_{t \in T}\{u_t \in \mathcal{A}'\} \le X_h$$

Similar to the previous cases, the proposed solution is principled and well-aligned with the optimization objective that SmartCrowd proposes. The solution involves only $|\mathcal{A}'| \times |T|$ variables, and our experimental results corroborate that it generates the output within reasonable latency.

## 5 Greedy Approximation Algorithms

The optimal algorithm presented in Section 4 may be very expensive regarding index building and mainte-

[6] If none of the workers in $\mathcal{A}'$ contributed to $t$, then $v'_t = v_t$.

nance time, since the ILP-based solution has an exponential computation time in the worst case. To expedite these steps, in this section we propose two approximate solutions, one randomized and one deterministic, which are both guaranteed to run in polynomial time and they have provable approximation factors under certain conditions. The randomized algorithm admits a better approximation factor than the deterministic one, when the objective function is only submodular. The deterministic algorithm requires both submodularity and monotonicity to admit the provable approximation factor, but it is computationally more efficient than its randomized counterpart. The randomized algorithm is presented in sub-section 5.1 and the deterministic in sub-section 5.2.

**Algorithm 2** Offline-CDEX-Randomized

**Input:** Workload $T = \{t_1, t_2, \ldots, t_l\}$, $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$, $X_h$

1: Fix parameters, $\delta, \delta^{'} \in [-1, 1]$. Start with $\mathcal{A} = \{A_1 = \{\}, A_2 = \{\}, \ldots A_l = \{\}\}$, no of elements $X = n \times X_h$.
2: Call `Offline-CDex-ApproxDeterministic` to get an estimate of value for optimal worker to task assignment, i.e., OPT.
3: For each element $u$ and set $A_t$, define $w_{A_t,\delta}(u) = E[f(R(A_t, \delta) \cup \{u\})] - E[f(R(A_t, \delta) \backslash \{u\})]$.
By repeated sampling, compute $\bar{w}_{A_t,\delta}(u)$, an estimate of $w_{A_t,\delta}(u)$ within a factor $\pm\frac{1}{(n \times X_h)^2}$ of OPT.
4: If there exists an $u \in X \backslash A_t$ such that $\bar{w}_{A_t,\delta}(u) > \frac{2}{(n \times X_h)^2} OPT$, include $u$ in $A_t$ and go to step 3.
5: If there exists an $u \in X \backslash A_t$ such that $\bar{w}_{A_t,\delta}(u) < -\frac{2}{(n \times X_h)^2} OPT$, exclude $u$ from $A_t$ and go to step 3.
6: Return a random set $R(A_t, \delta')$.
7: **return** the index set $\mathcal{I} = \{A_1, A_2, \ldots, A_l\}$.

## 5.1 Greedy C-DEX Randomized algorithm

### *5.1.1 C-DEX Randomized index building (Offline phase)*

`Offline-CDEX-Randomized:` This randomized approximation algorithm is an adaptation of the solutions proposed in [11]. [11] proposes its randomized algorithm for a single set (analogous to a single task in our case), whereas here we need to perform the assignment for a set of tasks.

The intuitive idea behind the algorithm described in [11] is to perform an "adaptive local search". It proceeds by locally optimizing a smoothed variant of the optimization function $f(S)$, obtained by biased sampling depending on $S$. The approach of locally optimizing a modified function has been referred to as "non-oblivious local search" [1] in the literature. For smoothing, the aforementioned work [11] uses a *multi-linear relaxation* [46] of the objective function. In particular, the algorithm in [11] starts with an empty set. For each element $x$, it computes the marginal gain of adding the element by computing multiple possible random sets with and without $x$. These steps are referred to as "smoothing". Then the element that has a marginal gain greater than a threshold value is added to the set. Similarly, if any element in the current solution has a marginal value less than a threshold it is dropped. This process is repeated until the objective function reaches a local optimum. In this randomized local search, the elements are sampled randomly with different probabilities. In other words, given a current solution $S$, the idea is to do a biased sampling based on whether an element is present in the set.

This algorithm [11] is adapted to our problem. It is easy to see that the items in [11] correspond to workers. Algorithm 2 contains the pseudo-code. Given the pool of tasks and workers, the workers are sampled randomly with different probabilities. We start with a set of null sets, i.e., $\mathcal{A} = \{A_1 = \{\}, A_2 = \{\}, \ldots A_l = \{\}\}$, where the number of sets equals the number of tasks. $\delta, \delta^{'}$ are two fixed parameters $\in [-1, 1]$. Theorem 3.6 in [11] suggests that we set $\delta$ as $1/3$ and $\delta'$ is chosen randomly to be $1/3$ with probability 0.9 and $-1$ with probability 0.1. We have adhered to this suggestion in our implementation. $R(A_t, \delta)$ denotes a random set of workers for a task $t$, where the workers in $A$ are sampled with probability $p = \frac{1+\delta}{2}$ and workers outside $A$ are sampled with probability $q = \frac{1-\delta}{2}$.

For each worker $u$, in an iteration, the weight of $u$ is computed as $w_{A,\delta,t}(u)$, which is the *marginal gain* of adding $u$ to the random set $R(A_t, \delta)$ in an expected sense (Step 3 of Algorithm 2). In other words, we construct a random set and see the marginal utility of adding this worker. We repeat this process numerous times to get an estimate of the worker's real weight, $\bar{w}_{A_t,\delta}(u)$. After that, worker $u$ is either included in $A_t$ or excluded from $A_t$ based on a certain probability check (Step 4 and 5 of Algorithm 2). Finally, a random set $R(A_t, \delta')$ is returned for a given $\delta'$, representing the assignment of a set of workers to task $t$ (Step 6 of Algorithm 2).

In order to decide whether to add a worker to the solution, the algorithm uses a probabilistic check that requires an estimate of the optimal solution. We use the same technique as [11] and use a *deterministic local search* algorithm (referred to as `Offline-CDex-ApproxDeterministic` as a subroutine for this purpose (Step 2 of Algorithm 2). Further, since a worker is allowed to be assigned to at most $X_h$ tasks, a total number of $X_h \times n$ elements (each element is a worker) are created.

**Algorithm 3** Offline-CDex-ApproxDeterministic

**Input:** Workload $T = \{t_1, t_2, \ldots, t_l\}$, $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$, $\mathcal{A} = \{A_1 = \{\}, A_2 = \{\}, \ldots A_l = \{\}\}$, $X_h$, $X = n \times X_h$

1: For each $A_t$, choose a single distinct worker $u$ from $X$ who maximizes $f(\{u\})$ and remove $u$ from $X$
2: If there exists an element $u' \in X \backslash A_t$, such that $f(A_t \cup \{u'\}) > (1 + \frac{\epsilon}{n^2}) f(A_t)$ , then $A_t = A_t \cup \{u'\}$. Go back to step 1.
3: If there exists an element $u' \in A_t$, such that $f(A_t \backslash u') > (1 + \frac{\epsilon}{n^2}) f(A_t)$, then $A_t = A_t \backslash u'$. Go back to step 1.
4: $\forall t \in T$, $v_t$ = maximum of $f(A_t)$ and $f(X \backslash A_t)$
5: **return** $\mathcal{V} = \sum_{t \in T} v_t$ as an estimate of OPT.

The subroutine `Offline-CDex-ApproxDeterministic` is called inside Offline-CDEX-Randomized to estimate the $\mathcal{V}$, i.e. OPT. This algorithm also runs in a greedy fashion, to increase the value $\mathcal{V}$ of our solution; in each iteration, it either includes a new element $u$ in $A_t$ or discards it from $A_t$. Whether an element would be added or discarded is based on the check that is described in Step-4 of Subroutine 3. This algorithm has an approximation factor of 1/3, which could be proved by directly using the results of [11].

**Theorem 5** `Offline-CDEX-Randomized` *has an approximation factor of* 2/5*, when* $Q_{t_j} = 0, \forall j \in \{1..m\}$.

*Proof* Sketch: Section 3.1.1 proves that $\mathcal{V}$ becomes submodular under the above-mentioned conditions. After that, the approximation factor follows directly from [11].

**Lemma 3** *The run time of algorithm* `Offline-CDEX-Randomized` *is polynomial, i.e.* $O(\frac{(X_h \times |\mathcal{U}|)^2}{\delta} \times |T|)$.

*Proof* Based on the previous result [11], the number of iterations per task is atmost $O(\frac{(X_h \times |\mathcal{U}|)^2}{\delta})$; after that, our result trivially follows.

*5.1.2 C-DEX Randomized index maintenance (Online phase)*

Akin to the offline scenario, we propose a randomized greedy approximation algorithm `Online-CDEX-Randomized` that is incremental and designed to ensure worker non-preemption.

*Replacing Workers:* After a task arrives if one or more of the assigned workers to this task are not available, an efficient greedy solution is proposed by selecting replacement workers from the available pool. This strategy leads to a provable approximation factor of 2/5, when $Q_{t_j} = 0, \forall j \in \{1..m\}$. `Online-CDEX-Randomized` works akin to `Offline-CDEX-Randomized`, except that it needs to find replacement workers for a single task $t$. Given a set of unavailable workers in $\mathcal{L}_i^t$, SmartCrowd considers the available set of workers $\mathcal{U}'$, and repeats Algorithm 2 considering only those workers for $t$ and their available bandwidth. The rest of the algorithm is akin to the one described earlier.

**Theorem 6** `Online-CDEX-Randomized` *admits an approximation factor of* 2/5*, when* $Q_{t_j} = 0, \forall j \in \{1..m\}$.

*Proof* Sketch: Our proof uses the submodularity property of $v_t$ as proved in Section 3.1.1 under these conditions.

Of course, unless the above conditions are satisfied, the above approximation factor does not theoretically hold.

**Lemma 4** *The run time of* `Online-CDEX-Randomized` *is polynomial, i.e.,* $O(\frac{(X_h \times |\mathcal{U}|)^2}{\delta})$.

*Addition of New Workers:* Our proposed randomized algorithm can be used to assign new workers to the tasks. This is similar in principle to the offline greedy randomized approximation algorithm described above. New workers must be assigned to the pre-computed indexes using Algorithm 2 (randomized solution) without disrupting the existing allocation of the current workers. However, in order to satisfy any theoretical guarantee, the objective function has to relax the quality threshold constraint (to satisfy submodularity). The run time complexity of the algorithm remains unaltered.

*Deletion of Workers:* The randomized approximation algorithm is adapted to handle worker deletions, akin to the greedy worker replacement strategy described above. It admits the exact same set of theoretical claims under similar conditions as described above.

*Updates of Worker Profile:* If the skill, wage, or acceptance ratio of a subset $\mathcal{A}'$ of workers gets updated, SmartCrowd first updates the respective *value* of the tasks (where these workers were assigned), by discounting the contribution of the workers in $\mathcal{A}'$. After that, the greedy randomized approximation algorithm `Offline-CDEX-Randomized` is adapted involving $\mathcal{A}'$ workers and $T$ tasks. It iteratively adds a worker in $\mathcal{A}'$ to a task in $T$ based on sampling as described in Algorithm 2, while satisfying the skill, cost, and number of workers per task constraint. Akin to `Offline-CDEX-Randomized`, this algorithm does not satisfy the 2/5 approximation factor unless $Q_{t_j} = 0, \forall j \in \{1..m\}$.

## 5.2 Greedy C-DEX Deterministic algorithm

### *5.2.1 C-DEX Deterministic index building (Offline phase)*

Next we describe the second approximation algorithm `Offline-CDEX-Deterministic`, which has a provable approximation factor when the function is submodular and monotonic and is more computationally efficient compared to its randomized counterpart.

Given the pool of tasks and workers, the algorithm iteratively adds a worker to a task such that the addition ensures the *highest marginal gain* in $\mathcal{V}$ in that iteration, while ensuring the quality, cost, and tasks-per-worker constraints. Imagine a particular instance of `Offline-CDEX-Deterministic` on Example 1 after the first iteration. After a single worker assignment (first iteration will assign one worker to one of the indexes), if only $u_1$ is assigned to $i^{t_1}$ and nobody to $i^{t_2}$ and $i^{t_3}$ yet, then the algorithm may select $u_6$ to assign to $i^{t_3}$ in the second iteration to ensure the highest marginal gain in $\mathcal{V}$.

**Theorem 7** `Offline-CDEX-Deterministic` *has an approximation factor of* $(1 - 1/e)$*, when* $Q_{t_j} = 0, \forall j \in \{1..m\}$ *and* $C_2 = 0$*.*

*Proof* Sketch: The proof directly uses the results of Section 3.1.2 (Theorem 4) and on the fact that the optimization function $\mathcal{V}$ becomes submodular and monotonic under the above-mentioned conditions. After that, the approximation factor follows directly from [38].

**Lemma 5** *The run time of algorithm* `Offline-CDEX-Deterministic` *is polynomial, i.e.,* $O(X_h \times |\mathcal{U}| \times |T|)$*.*

It is evident that `Offline-CDEX-Deterministic` is more efficient than its randomized counterpart `Offline-CDEX-Randomized`(referred to lemma 3).

### *5.2.2 C-DEX Deterministic index maintenance (online phase)*

*Replacing Workers:* The worker replacement strategy in this case is a deterministic greedy algorithm `Online-CDEX-Deterministic` that leads to a provable approximation factor when $Q_{t_j} = 0, \forall j \in \{1..m\}$ $C_2 = 0$. We describe its functionality next.

`Online-CDEX-Deterministic:` Given a set of unavailable workers in $\mathcal{L}_i^t$, SmartCrowd performs a simple iterative greedy replacement from the available pool of workers $\mathcal{U}'$. In a given iteration, the idea is to select the worker from the available pool, whose addition will give the *highest marginal gain in* $v_t$, and add her to $\mathcal{L}_i^t$. This iterative process continues until the cost constraint exceeds. This greedy algorithm is approximate in nature but admits a provable approximation factor under certain conditions.

**Theorem 8** `Online-CDEX-Deterministic` *admits an approximation factor of* $1 - 1/e$*, when* $Q_{t_j} = 0, \forall j \in \{1..m\}$ *and* $C_2 = 0$*.*

*Proof* Sketch: Our proof uses the monotonicity and submodularity property of $v_t$ as proved in Section 3.1.1 under these conditions.

Of course, unless the above conditions are satisfied, the above approximation factor does not theoretically hold.

**Lemma 6** *The run time of* `Online-CDEX-Deterministic` *is polynomial, i.e.,* $O(|\mathcal{U}'|)$*.*

*Addition of New Workers:* Again the proposed deterministic algorithm is used to assign new workers to the tasks similarly to the offline greedy deterministic approximation algorithm described above. New workers are assigned to the pre-computed indexes using the highest marginal gain of the deterministic solution without disrupting the existing allocation of the current workers. In order to satisfy any theoretical guarantee, the objective function has to satisfy both submodularity and monotonicity. The run time complexity of the algorithms remains unaltered.

*Deletion of Workers:* A deterministic approximation algorithm is proposed to handle worker deletions, similarly to the greedy worker replacement strategy described above. It admits the exact same set of theoretical claims under similar conditions.

*Updates of Worker Profile:* If the skill, wage or acceptance ratio of a subset $\mathcal{A}'$ of workers gets updated, SmartCrowd first updates the respective *value* of the tasks (where these workers were assigned) by discounting the contribution of the workers in $\mathcal{A}'$. After that the greedy solution is proposed as follows. A deterministic approximation algorithm is constructed that adapts `Offline-CDEX-Deterministic` involving $\mathcal{A}'$ workers and $T$ task. It iteratively adds a worker in $\mathcal{A}'$ to a task in $T$ based on the highest marginal gain in value, while satisfying the skill, cost and number of workers per task constraint. Akin to `Offline-CDEX-Deterministic`, this algorithm does not satisfy the $(1 - 1/e)$ approximation factor unless $Q_{t_j} = 0, \forall j \in \{1..m\}$ and $C_2 = 0$.

## 6 Clustering-based approximation algorithm: C-DEX$^+$

In this section we present a different approximate solution, which is called C-DEX$^+$ and it is based on an alternative index building idea. In C-DEX$^+$, the actual worker pool is intelligently *replaced* by a set of *Virtual Workers*, which are much smaller in count. SMARTCROWD uses the Virtual Workers and the same workload to pre-compute a set of indexes, referred to as C-DEX$^+$. The C-DEX$^+$ approach enables efficient pre-computation, as well as faster assignments from workers to tasks. C-DEX$^+$ is an approximate solution, and the quality of its approximation is hinged on how the *Virtual Workers* are created.

Intuitively, a Virtual Worker represents a set of "indistinguishable" actual workers, who are similar in skills and cost. For the simplicity of exposition, if we assume that in a given worker pool, there are 3 workers who posses exactly the same skill $s$ and cost $w$, then a single Virtual Worker $V$ could be created replacing those 3 with skill $s$ and cost $w$. Obviously, when there are variations in skills and costs of workers, the profile of $V$ needs to be defined conservatively - by taking the maximum cost of the individual workers as $V$'s cost, and the minimum of the individual worker's expertise per skill, as $V$'s skill. The formal definition of $V$ is:

**Definition 2** Virtual Worker V : V represents a set $n'$ of actual workers that are "indistinguishable". V is an $m+2$ dimensional vector, $\langle V_{s'_1}, V_{s'_2}, \ldots, V_{s'_m}, V_{w'}, |n'| \rangle$ describing expected skill, expected wage, number of actual workers in $V$, where, $V_{s'_i} = \min_{u \in n'} p_u \times u_{s_i}$, $V_{w'} = \max_{u \in n'} p_u \times w_u$.

Consider Example 1 again, if $u_2$ and $u_5$ are grouped together to form a Virtual Worker $V$, then $V = \langle 0.21, 0.18, 2 \rangle$.

It is apparent that the Virtual Workers help reduce the size of the optimization problem. The formal definition of C-DEX$^+$ is:

**Definition 3** (C-DEX$^+$) A C-DEX$^+$ $i^{t_V} = (\mathcal{P}_i^{t_V}, \mathcal{L}_i^{t_V})$ is a pair that represents an assignment of a set of Virtual Workers in $\mathcal{N}$ to a task $t$. $\mathcal{P}_i^{t_V}, \mathcal{L}_i^{t_V}$ are similar to $\mathcal{P}_i^t, \mathcal{L}_i^t$, defined using the Virtual Worker set $\mathcal{N}$ .

### 6.1 C-DEX$^+$ Design (offline phase)

We work in two steps: 1) Creating the Virtual Workers and 2) Designing the C-DEX$^+$.

*Creating Virtual Workers:* First, a set $\mathcal{N}$ of Virtual Workers is created, given $\mathcal{U}$. Intuitively, a Virtual Worker $V$ should represent a set of workers who are *similar* in their profile. In SMARTCROWD, Virtual Workers are created by performing multi-dimensional clustering [17] on $\mathcal{U}$, and considering a *threshold* $\alpha$ that dictates the maximum *distance* between any worker-pairs inside the same cluster. The size of the Virtual Worker set $\mathcal{N}$ clearly depends on $\alpha$, with a large value of $\alpha$ leading to smaller $|\mathcal{N}|$, and vice versa. Interestingly, this allows flexible design, as the appropriate trade-off between the quality and the run-time complexity could be chosen by the system, as needed. Formally, given $\mathcal{U}$ and $\alpha$, the task is to design a set of Virtual Workers, such that the following condition is satisfied:

$$\forall u, u' : u \in V, u' \in V, Dist(u, u') \leq \alpha$$

Our implementation uses a variant of Connectivity based Clustering [17] considering Euclidean distance to that end.

For example, if $\alpha = 0.25$, Example 1 will create $|\mathcal{N}| = 2$ Virtual Workers; $V_1$ with $\{u_1, u_2, u_3, u_5\}$ and $V_2$ with$\{u_4, u_6\}$; $V_1 = \langle 0.08, 0.18, 4 \rangle$ and $V_2 = \langle 0.3, 0.36, 2 \rangle$.

*Designing C-DEX$^+$:* For a Virtual Worker $V$ with $|n'|$ actual workers, a counter $\mathcal{C}_V$ is created stating the maximum assignments of $V$, i.e., $\mathcal{C}_V = |n'| \times X_h$. An ILP is designed analogous to Section 4 with $|\mathcal{N}|$ workers and all the tasks in $T$. Additionally, a total of $2|\mathcal{N}|$ constraints are added; one per $V$, stating that the maximum and the minimum allocation of $V$ are $\mathcal{C}_V$ and $(|n'| \times X_l)$, respectively.

**Lemma 7** *The optimization problem for C-DEX$^+$ involves only $|\mathcal{N}| \times |T|$ variables.*

Using the above lemma, it is easy to see that the ILP is likely to get solved faster for C-DEX$^+$, as it involves a smaller number of variables.

Example 1 gives 2 virtual workers $V_1, V_2$ when $\alpha = 0.25$. Two additional maximum allocation constraints will be added to the optimization problem, such that $\mathcal{C}_{V_1} = 4, \mathcal{C}_{V_2} = 2$. Therefore, the index-design problem with Virtual Workers could be solved for 3 tasks and 2 Virtual Workers, involving only $3 \times 2 = 6$ decision variables, instead of $6 \times 3 = 18$ variables that C-DEX has to deal with. While this solution is much more efficient compared to C-DEX, it may give rise to an approximation of the achieved quality (i.e. the value of the objective function $\mathcal{V}$), because the search space for the optimization problem gets further restricted with the Virtual Workers, leading to sub-optimal solution for $\mathcal{V}$.

**Algorithm 4** C-DEX$^+$Design Algorithm

**Input:** Workload $T$, $\mathcal{U}$, $\alpha$
1: Create $\mathcal{N}$, using $\mathcal{U}$ and $\alpha$.
2: Solve the C-DEX$^+$ Design ILP problem to get an assignment of Virtual Workers to the tasks.
3: **return** Index set $\mathcal{I}_V$.

Interestingly, our empirical results shows that this alternative solution is efficient and that the decline in the overall quality is negligible.

The output of the C-DEX$^+$ Design Algorithm is the set of task indexes $\mathcal{I}_V$ using virtual workers. Considering Example 1, $\mathcal{I}_V = \{i^{t_{V_1}}, i^{t_{V_2}}, i^{t_{V_3}}\}$. For task $t_1$, the algorithm created
$i^{t_{V_1}} = (\langle 0.38, 0.76, 1.08\rangle, \{V_1, V_1, V_2, V_2\})$, when $C_1 = C_2 = 0.5$. The individual worker-to-task assignment can be performed after that by a simple post-processing.

### 6.2 C-DEX$^+$ Maintenance (online phase)

Recall that the maintenance strategies are designed for 4 different scenarios, enforcing the worker non-preemption constraint.

*Replacing Workers:* C-DEX$^+$ designs a marginal ILP, involving task $t$ and all the Virtual Workers whose current $\mathcal{C}_V > 0$, akin to its C-DEX counterpart. Once the solution is achieved, individual worker assignments can be performed with a post-processing algorithm, in a round robin fashion, by keeping track of individual $L_V$'s.

*Addition of New Workers:* First, the existing Virtual Worker set $\mathcal{N}$ needs to get updated. Since $\alpha$ is predetermined, the new workers can be accommodated with incremental clustering, just by forming new clusters (i.e., creating new Virtual Workers) involving those additions, without having to re-create the entire $\mathcal{N}$ from scratch. After that, a smaller ILP is formulated, involving only the Virtual Workers that are affected by the updates and considering existing partial assignments, akin to Equation 3. We omit the details for brevity.

*Deletion of Workers:* The handling of worker deletion is akin to addition, in the sense that SMARTCROWD propagates these updates incrementally to the Virtual Worker set $\mathcal{N}$. To satisfy the pre-defined $\alpha$, it accounts for the remaining actual workers of each Virtual Worker $V$ that has at least one worker deletion. It reruns a smaller clustering solution involving only those remaining workers. As soon as $\mathcal{N}$ gets updated, the rest of the maintenance is exactly the same with what is discussed in Section 4.2 regarding deletion handling. We omit the details for brevity.

*Updates of Worker Profile:* Similarly, if SMARTCROWD gets to have an updated profile of some of the workers, it first updates the Virtual Workers set by solving a smaller clustering problem, akin to deletion. With the updated Virtual Workers set, the rest of the maintenance is the same as solving a marginal ILP involving only the updated Virtual Workers, similarly to the profile update maintenance discussed in Section 4.2.

## 7 Experimental Evaluation

We perform two different types of experiments. Real-data experiments are conducted involving 250 AMT[7] workers in three different stages. Synthetic-data experiments are conducted using an event-based crowd simulator.

### 7.1 Real-Data Experiments

The purpose of these experiments is to evaluate our approach in terms of feasibility and quality. We study *feasibility* since the current paid crowdsourcing platforms (like AMT) do not support KI-C task development and thus this is one of the first studies trying to optimize KI-C in such an environment. We study *quality* with the aim to measure the key qualitative axes of the knowledge produced by the hired workers.

Overall the study is designed as an application of collaborative news document writing. Specifically, workers hired through AMT are asked to produce documents on 5 diverse topics (KI-C tasks) of current interest: 1) Political unrest in Egypt, 2) NSA document leakage, 3) Playstation (PS) games, 4) All electric cars and 5) Global Warming. For simplicity and ease of quantification, we consider that each task requires one skill (i.e. expertise on that topic). To perform the assignment of workers to tasks, we compare three strategies. The first one is the C-DEX-Optimal, proposed by SMARTCROWD and presented in section 4. The second, namely `Benchmark`, is a rival strategy according to which the workers self-appoint themselves to articles after a skill-based pre-selection process, akin to how current paid platforms work. The third rival strategy, called `Online-Greedy`, assigns works to available tasks taking into account the workers' marginal utility - i.e. how much the worker's assignment improves the objective function of the task as defined in Equation 1 -

[7] Amazon Mechanical Turk, www.mturk.com

on each task; this is the adaptation of one of the latest state-of-the-art algorithms for online crowdsourcing task assignment [18].

The user study is conducted in 3 stages: i) worker profiling, ii) worker-to-task assignment and iii) task evaluation, as Figure 1 shows.

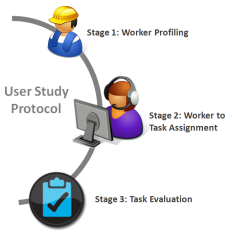



Fig. 1: Stages of the User Study

*7.1.1 Stage 1 - Worker Profiling*

In this stage, we hire 20 AMT workers per task (news topic), totaling 100 unique workers. The workers are informed that a subset of them will be invited (through email) in Stage 2 to collaboratively write a document on that topic.

**Learning Workers Skills:** We design pre-qualification tests to learn the skill of the workers. We design a questionnaire that comprises of 8 multiple choice questions per task, for which the ground truth is known. These questions are designed to assessing the workers' knowledge over facts related to the task. For example, for "Political Unrest in Egypt", we design questions, such as, *"What is the name of the place in Cairo where the protests took place?"* with possible answers: Tahrir Square, Mubarak Plaza, Al Azhar Square, or for the NSA leakage topic: *"Who is Adrian Lemo?* with possible answers: A computer hacker, A federal agent, Both). The skill of a worker is then calculated as the percentage of her correct answers.

**Learning Wage and Acceptance Ratio:** In the absence of historical data, we ask explicit question to learn the wage and acceptance ratio of the workers. For example, we ask each worker the following questions, Question to learn wage: "If you are asked to write a document on topic [X] within 150 words, how much would you typically charge?"
Question to learn acceptance ratio: "Out of 10 document editing tasks such as ours, how many will you typically accept?"
Figure 2 shows the quantification of worker profile distributions for the "Egypt" task. Worker profiles for the other topics exhibit similar distributions and are omitted for brevity. A strong positive correlation between the workers' skill and requested wage is also observed.

*7.1.2 Stage 2 - Worker-to-Task Assignment*

The set of 100 workers form our worker pool and we evaluate different task assignment strategies by comparing how these 100 workers are assigned to tasks. Specifically, in this stage, a subset (56 out of the 100) of the workers is selected and appointed to the tasks according to 3 worker-to-task assignment strategies: `C-DEX` (that corresponds to SmartCrowd), `Benchmark` and `Online-Greedy`, as presented above. Notice that the initially hired 100 workers comprise our original worker pool. Naturally, different task assignment algorithms only select a subset from there for each task.

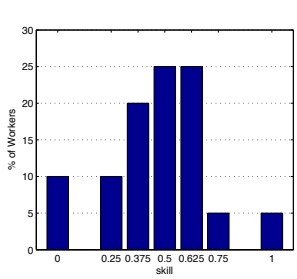
For `Benchmark`, we filter out workers with smaller than 0.5 skill (based on the pre-qualification test) and the rest of the workers self-appoint themselves to the task. For both `Benchmark` and `Online-Greedy`, any worker to task allocation that exceeds the maximum wage budget is also disregarded.

The minimum skill requirement per task is considered to be 1.8, the maximum wage \$2.0 and $C_1 = C_2 = 0.5$. The selected workers for each task are provided with a Google doc to collaboratively compose an article on the task's topic up to 150 words and in a time window of 24 hours. The workers are suggested to use the answers of the Stage-1 questionnaires, as a reference and/or starting point of their work. The workers are also asked to care for quality aspects of their article, such as language correctness and information completeness. The final outcome of this stage is a production of 3 documents per task (one document per assignment strategy), giving a total of 15 documents.

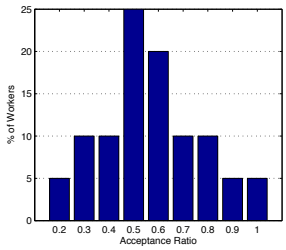
*7.1.3 Stage 3 - Task Evaluation*

KI-C evaluation is a delicate topic because it is subjective. An appropriate technique for such an evaluation is to leverage the *wisdom of the crowds*. This way a diverse and large enough group of individuals can accurately evaluate information to nullify individual biases and the herding effect. Therefore, in this stage we *crowdsource the task evaluation*. Each completed task (set of 3 documents) from Stage 2 is set up as a HIT in AMT, and 30 workers are assigned to evaluate it considering 5 key quality assessment aspects [6], without knowing the underlying task production algorithm. This is akin to obtaining the viewpoint of the average reader.

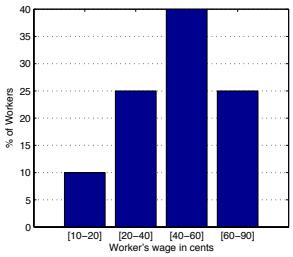
The results listed in Figure 3 indicate that the use of C-DEX indeed leads to more qualitative KI-C



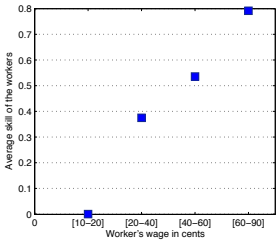

(a) Skill distribution




(b) Acceptance ratio distribution




(c) Wage distribution




(d) Strong positive correlation between worker skill and wage

Fig. 2: AMT worker profile distributions for the Egypt task

tasks, across *all* of the measured quality axes. During post-analyses of the results, we have observed that the documents produced by C-DEX have higher participation and edits from the participants, compared to the other two strategies. Indeed, both `Benchmark` and `Online-Greedy` take a myopic view to allocate workers to tasks and the allocation of workers to a task must stop as soon as the budget exceeds. Therefore, we observe that both of these strategies allocate lesser number of workers per task, which result in lesser edits, and lower overall quality consequently. These observations also implicitly corroborate the effectiveness of our additive skill model, where, workers indeed build on each others' contribution and improve the overall quality. Interestingly, we also observe that almost 90% of the time workers with high acceptance ratio indeed accept the tasks during Stage-2.

### 7.2 Synthetic Data Experiments

These experiments are conducted on an Intel core i7 CPU, 8 GB RAM machine. IBM CPLEX version 12.5.1 is used for solving the ILP. An event-based simulator is designed on Java Netbeans to simulate the crowdsourcing environment. All results are presented as the average of 3 runs.

*Simulator Parametrization:* The parameters presented below are chosen akin to their respective distributions, observed in our real AMT populations.
1. *Simulation Period* - We simulate the system for a time period of 10 days, i.e. 14400 simulation units, with each simulation unit corresponding to 1 minute.
2. *# of skills* - a total of $|\mathcal{S}|= 10$ skills are simulated. Unless otherwise stated, the default # of skills in a task is 1.
3. *# of Workers* - $|\mathcal{U}|= 10{,}000$.
4. *Profile of a worker* - $u_{s_i}$ in skill $s_i$ receives a random value from a normal distribution with the mean set to 0.5 and a variance 0.15. $w_u$ receives a random value from a normal distribution with a mean set to 0.5, variance 0.2. $p_u$ is also normal with a mean set to 0.5, variance 0.1.
5. *Tasks* - A normal variable with mean 15, variance 3 is multiplied with another normal random variable with mean 0.7, variance 0.15 to get $Q_{t_i}$. The former normal random variable is multiplied with a different normal random variable with mean 0.5, variance 0.2 to get $W_t$.
6. *Weights* - Unless otherwise stated, $C_1 = C_2 = 0.5$.
7. *Worker Arrival, Task Arrival* - Workers arrive following a Poisson process, with an arrival rate of $\mu = 10$/minute. Tasks arrive also based on a Poisson process with arrival rate $\kappa = 20$/minute.
8. *Workload* - Unless otherwise stated, the workload is designed with $10,000$ tasks.

*Implemented Algorithms:* `Benchmark`: It models a typical crowdsourcing environment, where workers are self-appointed to tasks, trying to maximize their individual profit. The algorithm also performs worker pre-filtering, similar to the pre-qualification tests used by today's crowdourcing platforms, allowing workers to undertake a certain task $t$ only if their skill is above 10% of the task's skill requirement $Q_{t_i}$.
`Online-Greedy`: As soon as a worker arrives, it finds from the available tasks the ones that pay more than the worker's minimum wage. Then it calculates the worker's marginal utility on the filtered tasks and suggests to the worker the task with the highest utility. This algorithm is an adaptation of one of the latest state-of-the-art strategies for online task assignment [18].
`Online-Optimal`: This approach assigns the incoming tasks to the available workers so as to maximize the objective function. In other words, as new tasks arrive, this algorithm optimally solves the ILP problem of Equation 1 (which in turn maximizes the objective function). However, when invoked it only uses the workers that are currently online. This algorithm acts

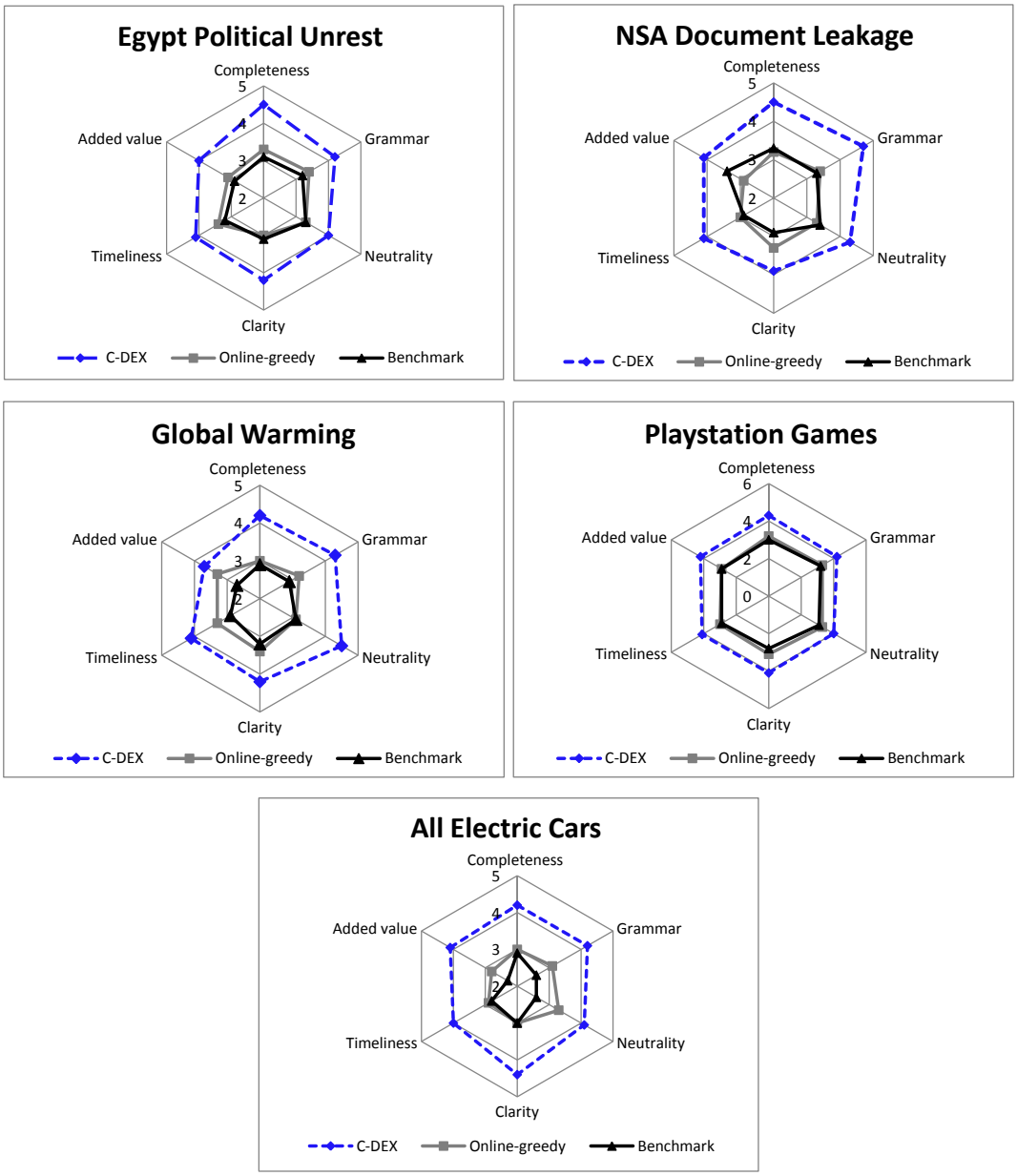



Fig. 3: Results of the real-user study on crowd journalism application tasks. The new articles written based on the C-DEX assignment are consistently better in quality compared to the rival assignment methods.

as a natural contrast to our proposed SmartCrowd algorithms (which all have an offline and an online phase) in that it performs optimal decisions based on the information available at that time.

`C-DEX:` generates the optimal solution for offline computation and online maintenance presented in Section 4.

`Offline-CDEX-Approx, Online-CDEX-Approx:` generates the approximate greedy solution of Section 5. We use the randomized variant (`Offline-CDEX-Randomized`) for the majority of the experiments as it had a better performance. However, we have also included a comparison of performance between the deterministic and the randomized variants.

`C-DEX`$^+$`:` generates the approximate clustering-based solution of Section 6.

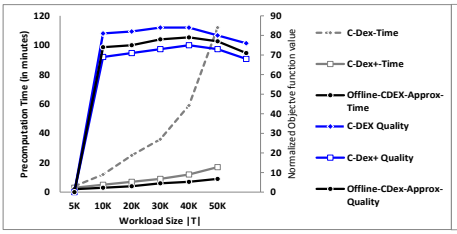



Fig. 4: Index Building Time and Quality varying workload

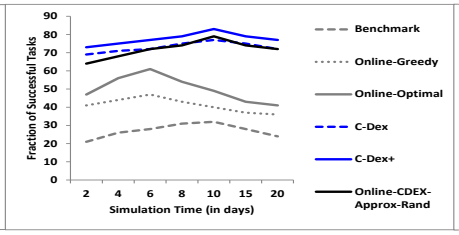



Fig. 5: Performance varying simulation time

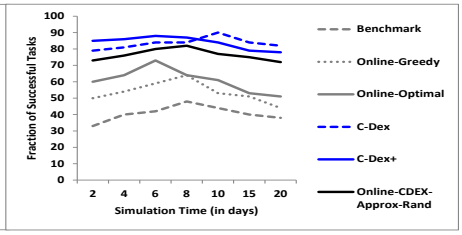



Fig. 6: Performance varying simulation time with no skill threshold and $C_2 = 0, X_l = 0$

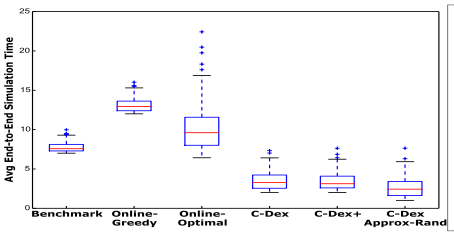



Fig. 7: Performance after entire simulation period

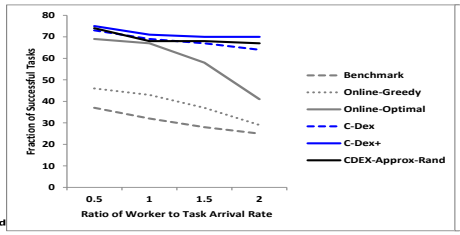



Fig. 8: Performance varying the ratio of task to worker arrival rate

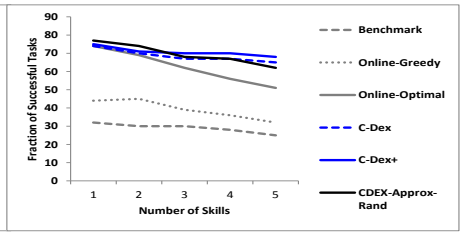



Fig. 9: Performance varying # of skills/task

*7.2.1 Performance Experiments*

We design experiments for both the offline phase (index building) and the online phase (index maintenance). Two measures are used: clock time for the index building and maintenance stages, and the fraction of successful tasks for the worker-to-task assignment stage ($\frac{\#\text{ of successful task assignments}}{\#tasks}$).

*Summary of Results:* The algorithms proposed in this paper `C-DEX`, `C-DEX`$^+$ and `Offline-CDEX-Approx-Randomized` require an additional preprocessing step for building an index. `C-DEX` is the most time-consuming as it performs an optimal worker to task allocation. `C-DEX`$^+$ and `Offline-CDEX-Approx-Randomized` require significantly less time as they only compute an approximate index. Nevertheless, we observe that this pre-processing cost is redeemed during the online phase, since our index-based strategies require substantially less time during runtime to perform the adaptive worker-to-task assignments, compared to our rival algorithms. We also observe a similar behavior during worker maintenance where the index-based strategies perform significantly better.

While `Online-Optimal` has a a performance (objective function value) comparable to `C-DEX` at the start (such as Figure 14), it degrades as the simulation ran longer. Making online decisions one task at a time (even if optimally per task), may indeed lead to undesirable outcome at the end. This is analogous to the classical difference between the online vs offline algorithms. By using the workload, `C-DEX` could be considered as an offline algorithm for the task-assignment problem. The results indicate that it is important to pro-actively do task assignment, so as to distribute workers and to take a holistic view of all tasks instead of optimizing one task at a time.

*Index Building (offline phase) Figure 4:* We vary the workload size of C-DEX, C-DEX$^+$ and `Offline-CDEX-Approx-Randomized` with $|\mathcal{U}| = 10,000$ and measure clock time for index computation (in minutes). Recall that C-DEX$^+$ needs to have the Virtual Worker set ($\mathcal{N}$) computed first. For that, our experimental evaluation sets $\alpha$ to 20-th percentile pair-wise Euclidean distance in ascending order, and observes that the computation time is within 2 minutes, resulting in $|\mathcal{N}| = 620$ Virtual Workers. The results are presented in Figure 4 (consider the primary Y-axis). Unsurprisingly, `Offline-CDEX-Approx-Randomized` is the fastest among the three alternatives, but C-DEX$^+$ is very comparable. Beyond $50,000$ tasks, C-DEX requires exorbitant time.

*Worker Replacement (online phase)* We compare the six implemented algorithms. The index-based algorithms (C-DEX,C-DEX$^+$ and `Online-CDEX-Approx-Randomized`) clearly win over the rest.

*Simulation period - Figures 5, 6, 7:* In Figures 5 and 6, we measure system performance (fraction of

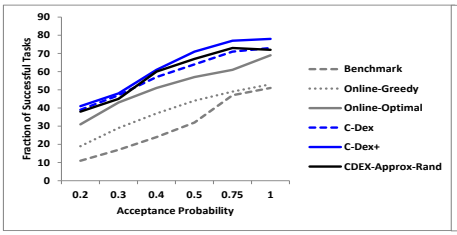



Fig. 10: Performance varying acceptance ratio

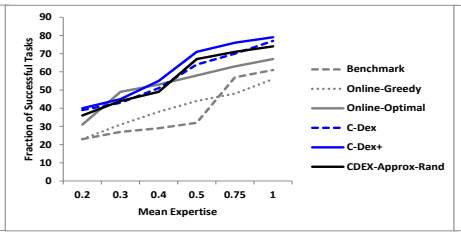



Fig. 11: Performance varying mean skill

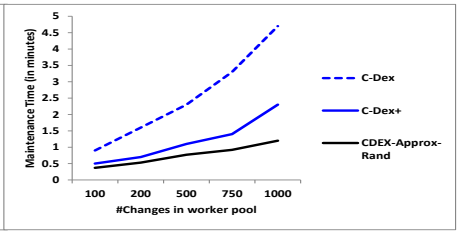



Fig. 12: Time for index maintenance varying # worker addition

successful tasks) throughout the simulation period at discrete intervals (every 2 days). Figure 6 captures the special case with $C_2 = 0, X_l = 0$ and zero skill threshold. Note that, under this condition `Online-CDEX-Approx-Randomized` has a provable approximation factor. We can observe that the proposed index-based strategies outperform the remaining ones significantly and that they maintain their throughput over the entire simulation period, while the other algorithms peak and then drop midway, as a result of their myopic worker-to-task assignment decisions that penalize the overall outcome. However, Figure 5 and 6 still depict a better-than-reality performance for some algorithms, since certain bad assignments are not counted as such due to the measurement discretization. For example, if a task comes at time unit 1, languishes until time 2388 before getting assigned, it will still count as a successful task. Figure 7 investigates this behavior by measuring average task end-to-end time, i.e. the difference in time between a task arrival and the time when a set of workers satisfying the task's quality/cost requirements have accepted to take it. This measurement is taken only for successful tasks and smaller is better. It can be observed that our proposed algorithms finish in less than 2 time units mainly because of our worker replacement strategy. The other algorithms including `Online-Greedy` take significantly more time. This justifies the necessity of pre-computation.

*Vary the ratio of task-to-worker arrival rate - Figure 8:* All algorithms perform well when the ratio of task arrival rate to worker arrival rate is small, because of the oversupply of workers. However, with high task arrival rate, the index-based strategies (`C-DEX` and its variants) outperform all the remaining solutions.

*Vary # skills/task - Figure 9:* As skills per task increase, the fraction of successful tasks decreases for all algorithms, since finding the right worker becomes harder in a high-dimensional task/worker setting. Nevertheless, the index-based strategies still manage to keep a steadily high performance, outperforming all remaining ones.

*Vary acceptance ratio - Figure 10:* With high acceptance ratio, performance improves in general, as workers become more predictable. The index-based strategies consistently outperform the others.

*Vary mean skill - Figure 11:* As expertise becomes scanty (i.e. low values of mean worker skill) `Benchmark` and `Online-Greedy` perform very poorly because they need to scan and seek more workers to reach the task skill threshold. The high performance on the other hand of C-DEX, C-DEX$^+$, and `Online-CDEX-Approx-Randomized` justifies that the optimization approach in SMARTCROWD is meaningful for knowledge-intensive tasks (since it is for these tasks that there may be fewer experts). Another interesting remark from this figure is the following: in case the expertise of the worker population is above average (i.e. above 0.5 out of 1), then our proposed techniques (C-DEX, C-DEX$^+$, and `Online-CDEX-Approx-Randomized`) are the ones to opt for. In case however the population expertise is below average (between 0 to 0.5 out of 1) then the `Online-Optimal` performs also well. This indicates that when expertise is low, even solving the ILP problem "naively" (i.e. in a purely online fashion like the `Online-Optimal` does) gives better results than current state-of-the-art and benchmark solutions. This advocates further the necessity of formulating KI-C optimization as an ILP problem, like we do in this paper.

*Worker Addition, Deletion, Update (online) - Figures 12 and 13* We vary the # of new workers, # of deleted workers,and # of workers with profile updates and measure the incremental maintenance time for C-DEX, C-DEX$^+$, and `Online-CDEX-Approx-Randomized`. The results for worker addition are presented in Figure 12. In Figure 13, we show the results of a more realistic scenario, where all three possible events could occur. To conduct this experiment, we alter the Poisson process for worker arrival, such that $\mu = 10$ now denotes an index maintenance event, which is either to add a new worker, modify the skill of a randomly chosen worker, or remove a randomly chosen worker. Results show

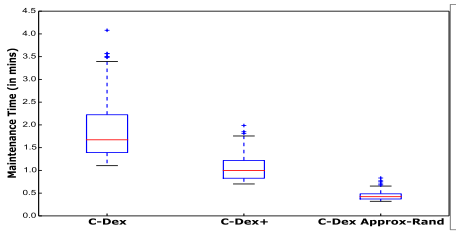



Fig. 13: Index maintenance with Worker Arrival, Deletion and Updation

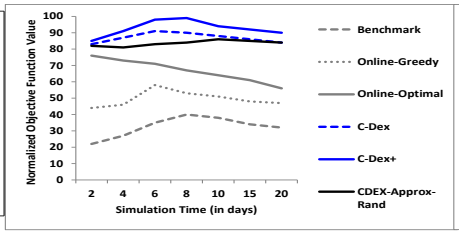



Fig. 14: Objective function varying simulation time

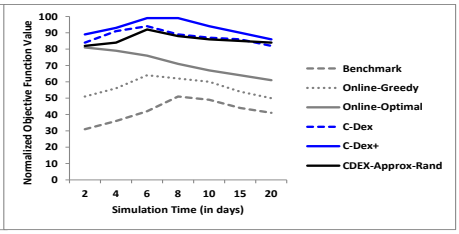



Fig. 15: Objective function varying simulation time with no skill threshold $C_2 = 0, X_l = 0$

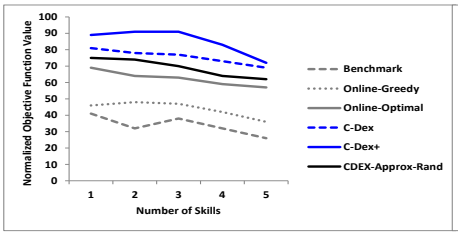



Fig. 16: Objective function varying # of skills/task

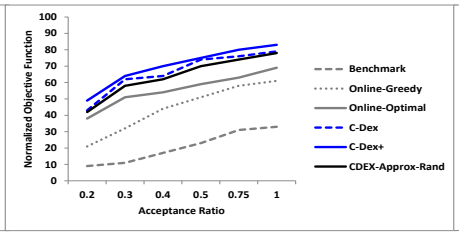



Fig. 17: Objective function varying acceptance ratio

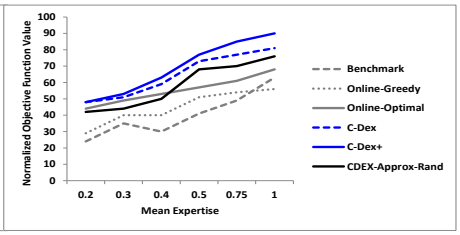



Fig. 18: Objective function varying mean skill

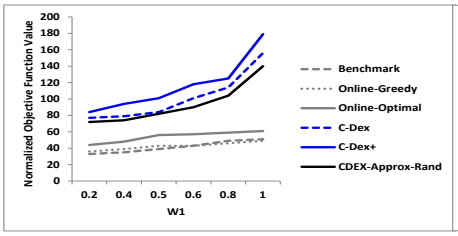



Fig. 19: Objective function varying $C_1, C_2$

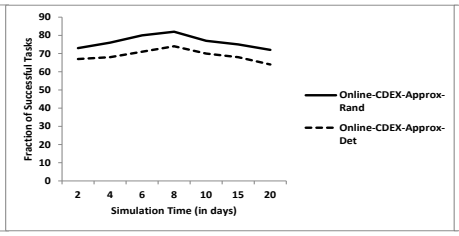



Fig. 20: Performance varying simulation time with no skill threshold $C_2 = 0, X_l = 0$

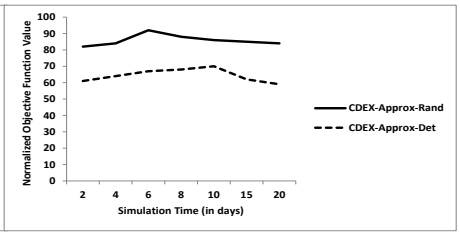



Fig. 21: Objective function varying simulation time with no skill threshold $C_2 = 0, X_l = 0$

that our incremental index maintenance techniques are efficient. However, the approximate solutions warrant higher efficiency compared to the optimal one.

### *7.2.2 Quality Experiments*

For the quality simulation experiments we measure the value of the normalized objective function.

*Index Building (offline phase)* The setting is akin to Section 7.2.1, but here we measure the objective function value instead. The results (consider the secondary Y-axis of Figure 4) demonstrate that both approximation algorithms C-DEX$^+$and `Offline-CDEX-Approx-Randomized` return high quality solutions that are comparable to their optimal counterpart C-DEX.

*Worker Replacement (online phase)*

*Simulation period - Figures 14 and 15* have similar settings to that of Figure 5 and 6. Our proposed index-based strategies significantly outperform the others throughout the period of the simulation. As expected, `Benchmark` performs the worst. `Online-CDEX-Approx-Randomized` returns higher quality in Figure 15 because the algorithm guarantees a provable approximation factor under these particular settings.

*Vary # skills-Figure 16:* The index-based strategies outperform all remaining ones, even for tasks that require multiple skills, similarly to Figure 9.

*Vary acceptance ratio - Figure 17:* The index-based strategies C-DEX, C-DEX$^+$, and `Online-CDEX-Approx-Randomized` outperform all the remaining ones, even with small mean worker acceptance ratio.

*Vary mean skill - Figure 18:* The index-based strategies consistently win over the rest, including the case where expertise is very scarce.

*Vary $C_1, C_2$ - Figure 19:* As expected, when $C_1$ increases, all algorithms seek to improve quality more than cost and task quality increases. The index-based solutions outperform the rest of their competitors with high $C_1$ (applications that require optimization over skills).

*Comparing Variants of* `Online-CDEX-Approx:` As our final experiment, we compare the performance of the two variants of approximation algorithms - deterministic (`Online-CDEX-Approx`) and randomized (`Online-CDEX-Approx-Randomized`). As mentioned in Section 5, these algorithms have provable approximation factors under certain conditions with the randomized one having a higher approximation factor when the objective function is submodular. Specifically, we perform our experiments for the special case where $C_2 = 0, X_l = 0$ and zero skill threshold, in which the objective function becomes submodular. As expected, Figure 20 shows that the randomized variant has a higher performance. In addition, as shown in Figure 21, the normalized objective function value is substantially better than the deterministic variant.

## 8 Related Work

To the best of our knowledge however, no work to-date formalizes the problem of KI-C optimization, examines it theoretically, or proposes principled solutions with performance guarantees to solve it. In this section, we discuss related work on traditional micro-task based crowdsorucing and team collaboration and describe the differences among them and our contributions.

**Micro-task based traditional crowdsourcing:** A growing number of crowdsourcing systems are available nowadays, both as commercial platforms (like AMT and CrowdFlower) or for academic use. Examples of applications include sentence translation, photo tagging and sentiment analysis, but also query answering (CrowdDB [12], Qurk [36], Deco [41], sCOOP, FusionCOMP, MoDaS, CyLog/Crowd4U), entity resolution(such as CrowdER [47]), planning queries [25], perform matching [48], or counting [35].

A common element shared across the above crowdsourcing systems is that the tasks that they handle are of *atomic/discrete-quality* nature. "Atomic" means that the task can be decomposed to a set of smaller units (e.g. a corpus of 10,000 images can be split to image-level micro-task), each of which is handled by individual workers (no collaboration). "Discrete-quality" means that each micro-task has one (or more) correct answer(s) out of a finite set of possible answers (e.g. an image either contains a person or not, a license plate contains one out of $n$ possible discrete possible numerical sequences). In other words each micro-task can be posited as a multiple-choice question or an n-ary classification problem [5,19]. Handling these tasks does not necessitate collaboration, but consensus. That is, many workers vote separately for each micro-task, and the task's "true value" (the correct out of the $n$ possible answers) is inferred by means of majority voting [22], cumulative voting [32], probabilistic generative modelling [37], or other techniques. The optimization problem in that case is to select the correct number of workers to identify the true values efficiently, with as low cost as possible (plurality optimization problem). Task recommendations to workers has been started to be considered by recent literature as a means to solve the plurality optimization problem for atomic/discrete quality tasks [26,4,40], while the vast majority of commercial systems still relies on "passive" quality assurance mechanisms like worker pre-filtering (using pre-qualification tests or "golden data" [10,23]) or task post-processing [49,42,16,4].

**Knowledge-intensive crowdsourcing (KI-C):** Knowledge-intensive crowdsourcing (KI-C) [29] handles tasks related to knowledge production, such as article writing, decision-making, science journalism. These tasks have two important differences from typical atomic/discrete-quality crowdsourcing discussed above. First KI-C tasks are less decomposable (writing a news article cannot be done by decomposing it to sentence level). Second, the quality of KI-C tasks is measured in a continuous, rather than a discrete quality scale. In other words KI-C tasks require an open and not a close-form answer, i.e. they do not have one correct answer out of a finite set of $n$ possible ones. The *non-decomposable/continuous-quality* nature of KI-C tasks necessitates collaboration among workers rather than their voting. It also changes the optimization problem, from a problem of plurality optimization ("should we hire another worker to correctly classify the task?) to a problem of worker-to-task assignment ("Which groups of workers should we hire to achieve that specific level of quality for the task?").

Up to now, no work or system optimizes the KI-C worker-to-task assignment process. Current systems decide task assignments either by allowing workers to self-appoint themselves to the tasks (like Wikipedia) or manually (like the oDesk platform[8] where the the per-

[8] https://www.odesk.com/

son who pays for the task has to personally select workers from a long candidate worker list). The first method (worker self-appointment) has negative effects on task quality with a long-tail of low-quality articles and a high cost in terms of user effort spent on certain articles [31]. The second method (manual assignment) leads to less-than-optimal allocations, bureaucracy and decision bottlenecks since the manual coordinator cannot possibly have an overview of the skills of every individual worker in a reasonable amount of time [24]. As for automated methods of system-wide worker-to-task assignments in KI-C, related works are unfortunately too few. Current studies simply acknowledge the need for more sophisticated strategies to handle task allocations [28,39,30] and highlight that the fact that task recommendations, when applied on typical atomic/discrete-quality crowdsourcing, have been shown to reduce delays in finding suitable tasks [51]. Related research also indicates that matching workers to tasks in wiki settings [34,14] can help increase the quality of the produced knowledge. Nevertheless, they do not formalize the problem of KI-C optimization, examine it theoretically or propose principled solutions to address it. Our contribution is one of the first ever attempts to address this gap.

**Team Collaboration:** Finally, our problem bears some resemblance with existing team formation problems in social networks (SN) [2], in the sense that here too users are grouped together with the purpose of collaboration on a set of tasks. There are however two critical differences: whereas SN-based team formation relies on user affinity within the social network, crowdsourcing entails a huge scale of diverse worker pool unknown to each other, who do not necessarily need the *synergy* of a "team" to work together (e.g., a Wikipedia-style of work can be used). Second, KI-C deals with unique challenges related to human factors in a dynamic environment, which is rarely seen for SN-based team formation.

## 9 Discussion

We examine the task assignment optimization of a new form of crowd work, namely knowledge-intensive crowdsourcing (KI-C), which focuses on knowledge production rather than on the accomplishment of simple human tasks. A lot more issues naturally emerge after this attempt, which need to be discussed and explored. These are related to integrating the human factor further into the task-assignment process, to examining more sophisticated worker collaboration models, to studying the peripheral processes on which KI-C optimization is based and other applications.

### 9.1 Human Factors and Alternative Modeling

We discuss additional human factors here. User motivation and its link to productivity is the first factor that comes to mind: studies (such as [50,27,20]) show that the correct incentives (not only monetary but also implicit ones like self-fulfillment, reputation, or personal interests) can play an important role in improving and sustaining the performance of crowd workers. One extension would be to enrich worker profiles accordingly and leverage information such as worker reputation and interests.

Skill improvement through training is a second human factor that can be very well coupled with our framework. Specifically, recent studies [9] show that for complex tasks, like creative design, worker engagement leads to increasing levels of expertise, because workers "learn by doing". Task assignment could adjust tasks to the "learning rate" of each worker.

Worker skill deterioration (e.g. due to boredom or fatigue) has also been observed, especially when workers are engaged over the same type of task too many times. Therefore, an important addition to the current assignment algorithms is task assignment variety [7] and advised micro-breaks [44], taking into consideration the personal tolerance levels of each worker in regards to task repetition.

**Alternative Wage Model:** it is also possible to devise alternate formalisms of the human factors such as skills, wage expectation, and acceptance ratio discussed in Section 2. For example, we could design a mechanism by which the wage expectation of a worker depends on the task being performed. This could model scenarios where the worker might want to charge a higher wage for tasks requiring advanced skills or requiring longer period for completion (for example, multiple hours or even days instead of mere minutes). Formally, we could define a function $f(u,t)$ that takes as input two parameters - user profile and task profile - and outputs the wage expectation for the given task, denoted by $w_{u,t}$. Interestingly, such generalization does not have any significant impact on the hardness of the problem which remains NP-Complete. It has only a minor impact in how the wage of the task is computed. In other words, it changes from $\sum_{u\in\mathcal{U}} u_t \times p_u \times w_u$ to $\sum_{u\in\mathcal{U}} u_t \times p_u \times w_{u,t}$ . As long as $w_{u,t}$ is a constant, no further changes are required.

### 9.2 Worker Collaboration

In this paper we assumed a wiki-like collaboration model, one in which workers collaborate on the task itself, editing each other's contributions, but not directly

interacting with one another, for example through discussion. This choice was made to enable hiring and assignment of workers that had not worked together before, through the popular platform of Amazon Mechanical Turk. Nevertheless, a setting where workers would be allowed to explicitly interact with each other could also be envisioned. Naturally, this setting makes the assignment process more complex since it involves the incorporation of additional parameters, like worker affinity, in the problem formulation. It also poses new constraints, for example that workers should not only have a collectively high expertise, but that they should also get along well one with the other, or that the assignment mechanism should effectively address risks emerging from user-to-user collaboration, like knowledge production bottlenecks and conflict resolution. We are currently examining specifically these directions.

A second topic in this line of research is the type of user roles assumed by the assignment mechanism. In our present work we implicitly assume one main worker role, i.e. the worker as a "contributor". More roles can be assumed, including workers as "reviewers" where workers evaluate each other's contributions. Note that this role has been partially implemented in our real-user study. Integrating it systematically into the worker-to-task assignment, e.g. by coupling the latter with a peer review process, will enable to further fine-tune the assignment process. Other roles that can be assumed, depending on the exact collaborative work setting and the specific nature of the crowdsourced tasks, include "task initiator", "team supervisor", "conflict mediator", "task workflow planner", "workflow controller" and "solution synthesizer" [9,30,52].

### 9.3 Other Components of KI-C

Our algorithms optimize worker-to-task assignment of KI-C. We thus rely on existing work to implement other components such as the *quantification of worker skills*, and the *evaluation of worker contributions*.

In regards to worker skill quantification, consider for instance the following: in our real-user study we map worker skills to numerical values by explicitly using a multiple-choice test that measures knowledge on our topics of interest. This method is adequate for our experimental proof-of-concept. However, for large-scale applications, this technique could induce extra cost and create a bottleneck between the time the worker joins the platform until she could start being assigned to tasks. Performing the quantification of worker knowledge in a universal and scalable fashion across platforms, for example by using explicit and machine learned evaluation [33] or implicit evaluation [8] of past worker contributions, could significantly minimize the cost of learning worker skills and thus help scale our approach. One very recent work that showcases how worker skill quantification can be achieved at large scale is [21].

In regards to task evaluation, we choose to crowdsource the assessment of each finished article, using a fixed number of worker-reviewers. Then, we assume a full confidence in these estimations, exactly as it is currently the case in many real-world systems like conference management ones. There are two issues here: one related to the subjectivity of the evaluations and the other related to the cost of the reviewers. Given that a person's evaluation of another person's contribution is always subjective, we need a sufficient number of reviewers to attain evaluations with acceptable confidence. On the other hand, adding too many reviewers can excessively increase the cost of article evaluations and this issue is crucial given the volume of tasks that our approach handles. It would be thus very interesting to see our method combined with works such as [26] that use statistical inference to obtain a good level of confidence over the estimation and subsequently stop the evaluation, as well as with works that use unsupervised statistical methods to estimate task quality for crowdsourcing tasks of unstructured response formats (like KI-C tasks) taking into account reviewer bias [3]. This would help answer questions such as: "what is the optimal number of reviewers needed to evaluate each specific task?", "what is the minimum evaluation confidence level to decide whether a certain task is above or below its quality threshold?", "what is the relative weight that each individual's assessment has?", as well as other related questions.

### 9.4 Beyond Collaborative Editing

In our real-user experiments we apply our framework to one specific KI-C application, that of citizen journalism. Based on this proof-of-concept, we can easily envision a meta-platform, on top of existing ones, for news coverage with quality and cost guarantees, competing with current centralized ones, like Reuters. It is our plan to examine building and making available such a meta-service framework, in the future.

Furthermore, it is crucial to notice that the knowledge-intensive tasks on which we focus are just one type of complex crowdsourcing task. Our methodology presents the advantage of being easy to adapt for the optimization of crowd performance on other complex task types. That is because most complex tasks, like idea generation, creative design or innovation seeding, are similar in nature to KI tasks, in that they also

require the collaboration of multiple workers of varying skills, while their quality is measured in a continuous rather than a discrete scale. Thus, one can envision many more complex-task applications that can profit from our approach.

**Fan-subbing Application:** As discussed in Introduction, there are other popular KI-C tasks such as *fan-subbing*[9], where a group of workers work together to translate a foreign movie (for example, translate a French movie in English). Fan-subbing requires multiple skills such as familiarity with two languages (source and target) in addition to other linguistic skills such as comprehension, transcription, etc. The skill aggregation is also additive for such an application, as the workers rectify the potential translation issues in output of other workers. The quality of a translation could be measured in a continuous scale using subjective measures and intuitively is proportional to the skills of workers who took part in the translation. The domain experts typically associate a desired quality per task. Workers are also incentivized, sometimes with actual money, which comprise of their wage. Naturally, a task is likely to be associated with a cost budget. We can now observe that this application could naturally be formulated using our framework.

For example, it would be very interesting to see how our framework compares and can help optimize current platforms, such as oDesk[10], which work with freelancers on creative tasks like marketing, web design, development or writing. Similarly, creative product design applications like Quirky.com[11] and social innovation ones like OpenIDEO.com[12], where users build on each other's ideas, can also significantly profit from our approach. Related to this, it is our plan to examine and expand our approach across platforms and on other types of complex tasks besides knowledge writing, like idea generation or creative design.

## 10 Conclusion

In this paper we propose SmartCrowd, a unified framework for optimizing worker-to-task assignment in knowledge intensive crowdsourcing (KI-C). KI-C is a new form of crowd work, which focuses on knowledge production rather than simple tasks and whose optimization in terms of quality and cost is critical to create value. Through SmartCrowd we formalize the KI-C problem and show that it can be mapped to an index design problem (C-dex), which we analyse theoretically and for which we propose optimal and approximate principled solutions. SmartCrowd relies on a set of pre-computed indexes, and maintains them adaptively to enable effective worker-to-task assignment. In addition, our framework integrates multiple human factors, such as worker expertise, minimum wage requirements and acceptance ratio, into the assignment process and it is flexible enough to be adapted to various KI-C applications. We validate SmartCrowd through extensive real-data and synthetic experiments, considering both quality and performance, and show that its algorithms can indeed guide the worker crowd towards superior knowledge production results. Multiple issues emerge, related to human factors, collaboration models, peripheral components and possible applications, which we discuss extensively. Overall, it is our belief that as crowdsourcing moves towards more complex tasks, and as knowledge production from the crowd becomes more and more important, SmartCrowd can help establish a basis for optimization, which future research efforts will find useful, apply and extend.

[9] http://en.wikipedia.org/wiki/Fansub
[10] https://www.odesk.com/
[11] https://www.quirky.com/
[12] http://openideo.com/

## References


1. Alimonti, P.: Non-oblivious local search for max 2-ccsp with application to max dicut. WG '97, pp. 2–14 (1997)
2. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Online team formation in social networks. In: WWW, pp. 839–848 (2012)
3. Baba, Y., Kashima, H.: Statistical quality estimation for general crowdsourcing tasks. KDD (2013)
4. Boim, R., Greenshpan, O., Milo, T., Novgorodov, S., Polyzotis, N., Tan, W.C.: Asking the right questions in crowd data sourcing. In: ICDE (2012)
5. Bragg, J..M., Weld, D.S.: Crowdsourcing multi-label classi cation for taxonomy creation. In: HCOMP (2013)
6. Chai, K., Potdar, V., Dillon, T.: Content quality assessment related frameworks for social media. ICCSA '09
7. Chandler, D., Kapelner, A.: Breaking monotony with meaning: Motivation in crowdsourcing markets. Journal of Economic Behavior & Organization **90**(0) (2013)
8. Dalip, D.H., Gonçalves, M.A., Cristo, M., Calado, P.: Automatic assessment of document quality in web collaborative digital libraries. JDIQ **2**(3) (2011)
9. Dow, S., Kulkarni, A., Klemmer, S., Hartmann, B.: Shepherding the crowd yields better work. CSCW (2012)
10. Downs, J.S., Holbrook, M.B., Sheng, S., Cranor, L.F.: Are your participants gaming the system?: screening mechanical turk workers. CHI '10
11. Feige, U., Mirrokni, V.S., Vondrák, J.: Maximizing non-monotone submodular functions. In: FOCS (2007)
12. Feng, A., Franklin, M.J., Kossmann, D., Kraska, T., Madden, S., Ramesh, S., Wang, A., Xin, R.: Crowddb: Query processing with the vldb crowd. PVLDB **4**(12)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness (1979)
14. Goel, G., Nikzad, A., Singla, A.: Allocating tasks to workers with matching constraints: Truthful mechanisms for crowdsourcing markets. WWW (2014)

15. Goemans, M.X., Correa, J.R. (eds.): *Lecture Notes in Computer Science*, vol. 7801. Springer (2013)
16. Guo, S., Parameswaran, A.G., Garcia-Molina, H.: So who won?: dynamic max discovery with the crowd. In: SIGMOD, pp. 385–396 (2012)
17. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2000)
18. Ho, C.J., Vaughan, J.W.: Online task assignment in crowdsourcing markets. In: AAAI (2012)
19. van der Hoek, W., Padgham, L., Conitzer, V., Winikoff, M. (eds.): IFAAMAS (2012)
20. Hossain, M.: Crowdsourcing: Activities, incentives and users' motivations to participate. In: ICIMTR (2012)
21. Ipeirotis, P., Gabrilovich, E.: Quizz: Targeted crowdsourcing with a billion (potential) users. In: WWW (2014)
22. Ipeirotis, P.G., Provost, F., Wang, J.: Quality management on amazon mechanical turk. HCOMP (2010)
23. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decis. Support Syst. **43**(2) (2007)
24. Joyce, E., Pike, J.C., Butler, B.S.: Rules and roles vs. consensus: Self-governed deliberative mass collaboration bureaucracies. American Behavioral Scientist **57**(5) (2013)
25. Kaplan, H., Lotosh, I., Milo, T., Novgorodov, S.: Answering planning queries with the crowd. In: PVDLB (2013)
26. Karger, D.R., Oh, S., Shah, D.: Budget-optimal task allocation for reliable crowdsourcing systems. CoRR **abs/1110.3564** (2011)
27. Kaufmann, N., Schulze, T., Veit, D.: More than fun and money. worker motivation in crowdsourcing-a study on mechanical turk. In: AMCIS (2011)
28. Kittur, A., Lee, B., Kraut, R.E.: Coordination in collective intelligence: The role of team structure and task interdependence. CHI (2009)
29. Kittur, A., Nickerson, J.V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., Horton, J.: The future of crowd work. In: CSCW '13 (2013)
30. Kulkarni, A., Can, M., Hartmann, B.: Collaboratively crowdsourcing workflows with turkomatic. CSCW '12
31. Lam, S.T.K., Riedl, J.: Is wikipedia growing a longer tail? GROUP '09 (2009)
32. Lee, S., Park, S., Park, S.: A quality enhancement of crowdsourcing based on quality evaluation and user-level task assignment framework. In: BIGCOMP (2014)
33. Lykourentzou, I., Papadaki, K., Vergados, D.J., Polemi, D., Loumos, V.: Corpwiki: A self-regulating wiki to promote corporate collective intelligence through expert peer matching. Inf. Sci. **180**(1) (2010)
34. Lykourentzou, I., Vergados, D.J., Naudet, Y.: Improving wiki article quality through crowd coordination: A resource allocation approach. Int. J. Semantic Web Inf. Syst. **9**(3), 105–125 (2013)
35. Marcus, A., Karger, D., Madden, S., Miller, R., Oh, S.: Counting with the crowd. In: PVLDB (2013)
36. Marcus, A., Wu, E., Karger, D., Madden, S., Miller, R.: Human-powered sorts and joins. PVLDB. (2011)
37. Matsui, T., Baba, Y., Kamishima, T., Hisashi, K.: Crowdsourcing quality control for item ordering tasks. In: HCOMP (2013)
38. Nemhauser, G., Wolsey, L., Fisher, M.: An analysis of approximations for maximizing submodular set functionsi. Mathematical Programming (1978)
39. O'Mahony, S., Ferraro, F.: The emergence of governance in an open source community. Academy of Management Journal **50**(5) (2007)
40. Parameswaran, A.G., Garcia-Molina, H., Park, H., Polyzotis, N., Ramesh, A., Widom, J.: Crowdscreen: algorithms for filtering data with humans. In: SIGMOD (2012)
41. Park, H., Widom, J.: Query optimization over crowdsourced data. In: VLDB (2013)
42. Ramesh, A., Parameswaran, A., Garcia-Molina, H., Polyzotis, N.: Identifying reliable workers swiftly. Technical report (2012)
43. Roy, S.B., Lykourentzou, I., Thirumuruganathan, S., Amer-Yahia, S., Das, G.: Crowds, not drones: Modeling human factors in interactive crowdsourcing. In: DBCrowd (2013)
44. Rzeszotarski, J.M., Chi, E., Paritosh, P., Dai, P.: Inserting micro-breaks into crowdsourcing workflows. In: HCOMP. AAAI (2013)
45. Soler, E.M., de Sousa, V.A., da Costa, G.R.M.: A modified primal-dual logarithmic-barrier method for solving the optimal power flow problem with discrete and continuous control variables. European Journal of Operational Research **222**(3) (2012)
46. Vondrák, J.: Symmetry and approximability of submodular maximization problems. FOCS (2009)
47. Wang, J., Kraska, T., Franklin, M.J., Feng, J.: Crowder: Crowdsourcing entity resolution. PVLDB (11)
48. Wang, J., Li, G., Kraska, T., Franklin, M.J., Feng, J.: Leveraging transitive relations for crowdsourced joins. In: SIGMOD Conference, pp. 229–240 (2013)
49. Whitehill, J., Ruvolo, P., Wu, T., Bergsma, J., Movellan, J.: Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In: NIPS (2009)
50. Yu, L., André, P., Kittur, A., Kraut, R.: A comparison of social, learning, and financial strategies on crowd engagement and output quality. CSCW (2014)
51. Yuen, M.C., King, I., Leung, K.S.: Task recommendation in crowdsourcing systems. CrowdKDD (2012)
52. Zhang, H., Horvitz, E., Miller, R.C., Parkes, D.C.: Crowdsourcing general computation. ACM CHI 2011 Workshop on Crowdsourcing and Human Computation