Efficient and adaptive incentive selection for crowdsourcing contests

Nhat Van-Quoc Truong[1] ⓘ · Le Cong Dinh[1] · Sebastian Stein[1] · Long Tran-Thanh[2] · Nicholas R. Jennings[3]

Abstract

The success of crowdsourcing projects relies critically on motivating a crowd to contribute. One particularly effective method for incentivising participants to perform tasks is to run contests where participants compete against each other for rewards. However, there are numerous ways to implement such contests in specific projects, that vary in how performance is evaluated, how participants are rewarded, and the sizes of the prizes. Also, the best way to implement contests in a particular project is still an open challenge, as the effectiveness of each contest implementation (henceforth, *incentive*) is unknown in advance. Hence, in a crowdsourcing project, a practical approach to maximise the overall utility of the requester (which can be measured by the total number of completed tasks or the quality of the task submissions) is to choose a set of incentives suggested by previous studies from the literature or from the requester's experience. Then, an effective mechanism can be applied to automatically select appropriate incentives from this set over different time intervals so as to maximise the cumulative utility within a given financial budget and a time limit. To this end, we present a novel approach to this *incentive selection problem*. Specifically, we formalise it as an online decision making problem, where each action corresponds to offering a specific incentive. After that, we detail and evaluate a novel algorithm, HAIS, to solve the incentive selection problem efficiently and adaptively. In theory, in the case that all the estimates in HAIS (except the estimates of the effectiveness of each incentive) are correct, we show that the algorithm achieves the regret bound of $\mathcal{O}(\sqrt{B/c})$, where B denotes the financial budget and c is the average cost of the incentives. In experiments, the performance of HAIS is about 93% (up to 98%) of the optimal solution and about 9% (up to 40%) better than state-of-the-art algorithms in a broad range of settings, which vary in budget sizes, time limits, numbers of incentives, values of the standard deviation of the incentives' utilities, and group sizes of the contests (i.e., the numbers of participants in a contest).

Keywords Incentive · Crowdsourcing · Online decision making

1 Introduction

Crowdsourcing has emerged as an efficient approach for obtaining solutions to a wide variety of problems by engaging a large number of Internet users from many places in the world [2–7]. Crowdsourcing is attractive to organisations and companies not only because it provides cheap labour, but also because it helps to solve problems quickly and effectively [8–10]. To cope with this increased number of crowdsourcing campaigns and the desire to run them quickly, there is growing interest in building autonomous agents to run crowdsourcing projects [11–21].

A key challenge in these settings is that the success of crowdsourcing projects relies critically on motivating a crowd to contribute [2, 10, 22]. Given this, contests[1] have been shown to be an efficient approach in these projects to motivate a crowd, as they are effective and cheap. Actually, by rewarding participants in a contest, task requesters do not necessarily have to pay for every task completed as in other types of financial rewarding schemes, such as

---

This paper is a significant extension of our paper published as an extended abstract at AAMAS 2018 [1].

✉ Nhat Van-Quoc Truong
   n.truong@soton.ac.uk

Extended author information available on the last page of the article.

---

[1]We use the term *contest* in a broad sense to refer to any situation in which participants exert effort to submit tasks for prizes, which are provided based on relative performance. The prizes can be tangible rewards, points, or positions on a leaderboard. Thus, all-pay auctions, lotteries, and leaderboards are considered contests for the purpose of this paper.

paying for performance [23] or using bonuses [24, 25]. Indeed, they have to pay mainly for a certain number of participants, e.g., the top two who have completed the most tasks or the top participant who has completed the tasks with the highest quality. 99 Designs (www.99designs.com), TopCoder (www.topcoder.com), and Taskcn (www.taskcn.com) are some well-known crowdsourcing platforms that use contests to attract participants.

Nevertheless, the effectiveness of the contests is likely to be different between crowdsourcing projects based on specific properties of those projects, such as the project purpose (e.g., building data for scientific studies or collecting data for a company), the task nature (e.g., interesting or boring), or the participant community (e.g., the extent to which they are in contact with each other or the extent to which the information of a participant is exposed to the others in the community). These differences reflect the fact that participants have different motivations [26–28]. Additionally, the implementation of the contests can make a significant difference in performance [29]. In more detail, each contest has a certain number of *parameters*, such as the base payment (a fixed payment for every participant), the group size (the maximum number of participants in a contest) and the amount of prize money for the best participant. For ease of presentation, we refer to a contest corresponding to specific parameter values as an *incentive*.[2]

Furthermore, currently on many crowdsourcing platforms such as Amazon Mechanical Turk (www.mturk.com), appen (https://appen.com/), and Clickworker (www.clickworker.com), the requesters can manage the tasks (e.g., creating a task with descriptions or uploading related data for a task) and the submissions (e.g., downloading the submissions from the participants or sending bonuses to participants with high quality submissions) in an autonomous manner using a programmable Application Programming Interface (API). This makes it possible to build autonomous agents to monitor and adaptively switch incentives when appropriate. Indeed, it is inconvenient or practically impossible in many cases to switch between incentives manually to identify

the best one. Therefore, finding an appropriate way for an autonomous agent to select an effective incentive in a crowdsourcing project is a key problem. We refer to this as the *incentive selection problem* (ISP) [1, 20].

As mentioned above, the effectiveness of the incentives in a specific crowdsourcing project is unknown in advance. Thus, in order to utilise the most effective one (i.e., exploit), the agent has to try each incentive several times to evaluate its respective effectiveness (i.e., explore). Given this need to balance exploitation and exploration, budgeted multi-armed bandits (budgeted MABs) are a suitable approach for this problem [30, 31]. In more detail, this approach models the problem as a machine with k arms (corresponding to k incentives), pulling an arm (providing the corresponding incentive to a group of participants) incurs a fixed cost (attached to the arm) and delivers a utility (e.g., the number of tasks completed) drawn from an unknown stochastic distribution. The objective in a budgeted MAB problem is to find a pulling policy (how many times each arm is pulled at each time step) that maximises the expected total utility within a given budget (e.g., £500).

A number of algorithms have been proposed to solve the budgeted MAB problem [18, 30–32]. However, these algorithms are not designed to work with the time budget (i.e., the deadline) of the ISP. Moreover, they do not consider the *group-based nature* of the incentives we consider here (i.e., contests); that is, after pulling an arm, we receive the performance of all the individuals in the corresponding contest group (i.e., a number of data points) rather than only the overall performance of the whole group (i.e., one data point). Thus, as we will show in Section 6, they are not efficient when dealing with the ISP. To illustrate the importance of the group-based nature, consider the two cases when the group size is 5 (i.e., 5 participants per contest) and 20 respectively. Current MAB algorithms would not treat these cases differently. However, the latter clearly provides us with more information on each pull (as it has more samples, i.e., participants). As a result, the second case requires fewer pulls of exploration in order to achieve the same level of understanding of the participants' performance (e.g., after 5 pulls of an arm in the second case, we have effectively sampled the performance of 100 individuals, but would require 20 pulls of an arm in the first case to reach that sample size). Hence, it is necessary to consider the group-based nature to determine the appropriate numbers of pulls for the arms. This is a non-trivial extension that requires new bandit algorithms.

In order to address this gap, in this paper we introduce an algorithm to deal with the ISP. The ultimate purpose of this work is to build an autonomous agent that can automatically and effectively select the right incentives, so that we can easily deploy projects on crowdsourcing platforms by using the provided APIs. To this end, our main contributions are:

---

[2]We can see that two incentives can be different in the parameters or the parameter values For example, we might have the following three incentives. Incentive 1 corresponds to contests where the base payment (the first parameter) is £1.00, the group size (the second parameter) is 10, and the amount of prize money for the best participant (the third parameter) is £5.00. Incentive 2 is the same as incentive 1 except the base payment is £1.50. Incentive 3 is the same as incentive 1 except it has one more parameter, the amount of prize money for the second participant is £5.00. In this example, incentives 1 and 2 have the same parameters but correspond to different values of the parameters. Also, incentive 3 has one parameter more than incentives 1 and 2.

Although the incentives focused on in this paper relate to contests, the problem stated and the algorithms discussed can be used with any other types of incentive in the literature, such as paying for performance or using bonus payments. Hence, to keep the problem general, we use the term *incentives* instead of *contests*.

(1) We formalise the ISP and then introduce HAIS, a novel adaptive algorithm to solve the ISP effectively by utilising the limited financial and time budgets while considering the group-based nature of the incentives. Specifically, HAIS is designed to have (a) a better exploration-exploitation strategy together with (b) an efficient way of using the time budget in the exploitation phase, and (c) an effective approach for spending more of the budget on highly effective incentives in the exploration phase.

(2) We theoretically show that in the case all estimates conducted in HAIS (except the estimates of the incentives' effectiveness) are correct, the algorithm benefits from a regret bound of $\mathcal{O}(\sqrt{B/c})$, where B denotes the financial budget and c is average cost of the incentives.

(3) We empirically demonstrate in synthetic environments that HAIS is more effective compared to the state-of-the-art approaches in an extensive series of simulations. Specifically, the performance of HAIS is about 93% (up to 98%) of the optimal solution and about 9% (up to 40%) better than state-of-the-art benchmarks.

The paper is organised as follows: next we discuss related work in Section 2. We then describe the ISP as a batched 2d-budgeted group-based MAB problem in Section 3. After that, we introduce HAIS in Section 4. We present a theoretical analysis of the algorithm by inspecting its regret bound in Section 5 and we conduct an empirical evaluation of the algorithm by running simulations in Section 6. We conclude in Section 7.

2 Related work

Much work has taken a game-theoretic approach to investigate the optimal (or efficient) design of contests in general and crowdsourcing contests in particular. Such work often tries to answer the questions of how to distribute the prizes (number of prizes and their values) in contests [33–41]. However, applying this body of research to building efficient contests for real-world crowdsourcing projects is still challenging since (1) these studies are based on the rationality assumption [3], whereas real participants in crowdsourcing might be partly rational or indeed irrational, as they might lack information, knowledge, or time; and (2) the studies do not consider other factors related to the participants' intrinsic motivation that might affect their behaviour, such as the project purpose, the task nature, or the participant community, as described in Section 1.

An alternative approach to deal with providing appropriate incentives is to design incentives that are thought to be effective based on previous studies and then empirically select the most effective one. Specifically, the above-mentioned studies can be used to design several contest implementations (i.e., candidate incentives). Other relevant studies in psychology, sociology, or computer science (e.g., [23, 26, 27, 42, 43]) can also be used for this task as they help us better understand possible interactions between the factors (e.g., the incentives, the level of autonomy and interestingness of the tasks, or the purpose of the projects) related to motivations of the participants. Then, based on the proposed candidates, an adaptive approach can be used to identify the most effective candidate efficiently.

Following this direction, as noted in Section 1, multi-armed bandits (MABs)(mab) are a promising approach. Some algorithms are shown to be robust in many cases, such as Exp3 [44] and Thompson Sampling [45]. Yet, as they do not consider the time and financial budgets, these algorithms cannot be used to solve the ISP. For example, with Exp3 and Thompson Sampling, in a time period, they draw an arm based on estimated probabilistic models corresponding to the arms. This is effective when the time budget is much larger than the number of arms. However, this is often not the case in the ISP. Therefore, budgeted MABs (MABs where the overall budget is limited, [30])(budgeted-mab) are a better approach. A number of studies considering budgeted MABs have been conducted. Specifically, the (financial) budget-limited MAB was first introduced by [30], where the number of times an arm can be pulled (in both exploration and exploitation phases) is constrained by a single budget B (without a deadline). Their algorithm (Budget-limited ϵ-first, or ϵ-first for short) spends ϵB (where ϵ is specified in advance, e.g., 0.3) for sequentially pulling the arms in the exploration phase and $(1 - \epsilon)B$ for pulling the arms with the highest estimated outcomes in the exploitation phase. [18] consider the uniform pulling approach of ϵ-first and argue that it might be inefficient in some cases when some ineffective arms can easily be identified and eliminated. From this argument, they develop three algorithms with three different approaches for eliminating the ineffective arms (l-split, PEEF, and SOAAv). Simulations on the trust problem of a supply chain [4] show these algorithms are effective, especially SOAAv with its adaptive approach. Taking a different approach, [31] use the idea of upper-confidence bounds from [46] to build an algorithm called Fractional KUBE, or fKUBE for short, that combines exploration and exploitation in one process.

However, as we will show in Section 6, the algorithms developed by [30, 31] and [18] are not efficient when dealing

---

[3]The assumption is that the human participants act in a fully rational manner.

[4]In this problem, a supplier can be considered as a node in a tree and each supplier faces a different MAB problem of choosing the most trustworthy sub suppliers.

with the ISP. First, regarding the group-based nature of the incentives (i.e., arms), the performance of these algorithms can vary significantly in different group sizes (i.e., the number of participants in a contest). For example, with ϵ-first, in the exploration phase, it pulls the arms evenly with a given budget, so the arms with smaller group sizes are explored less than larger group sizes. Indeed, as the total number of participants in each arm is the group size of the arm times the number of times this arm is pulled, when the arms are pulled the same number of times, the arm with a smaller group size has fewer participants. This will affect its performance when the group sizes of the best arms are small compared to those of the worst arms, as the best arms will be pulled less than the others in the exploration phase. Second, regarding the time limit, as they are designed to work with an unlimited deadline, their performance can drop significantly when running under strict time constraints. For example, with fKUBE, as it spends only one round to obtain initial estimates of the arms' effectiveness, when the deadline is tight, pulling the current best arm once in a period is too slow to identify the real best arm before the deadline. Third, regarding the exploration-exploitation balance, they do not have an effective and adaptive mechanism for distributing the financial budget across the two phases. For instance, with ϵ-first, it is not easy to specify an appropriate value for ϵ in advance, when there is little information about the performance of participants in the projects. Also, when differences in the effectiveness of the incentives are low (i.e., it is difficult to differentiate the incentives' effectiveness), the elimination mechanism of SOAAv might not be effective and the exploitation-exploration process of fKUBE might be slow in identifying the best arm. Despite these shortcomings, each algorithm also has its own strength, thus they are still good candidates for the ISP. Therefore, we implement these algorithms (with some modifications for the time constraint, when possible) to not only evaluate their performance but also to benchmark our new algorithm.

The work of [32] approaches budgeted MABs more generally by dealing with multi-dimensional bandits (each dimension corresponds to a resource, such as financial budget or time budget). However, their algorithms (PDBwK and BalanceBwK) cannot be applied to the ISP because the resources in their model cannot be shared, whereas in the ISP, the time can be shared, i.e., pulling one or more arms several times (providing one or more incentives to several groups) can happen in a given time period. Recently, [12] have attempted to use MABs to deal with the ISP. However, their model is difficult to use in practice, as they do not consider the time constraint (i.e., the tasks are provided one by one) and their algorithms are not adaptive (i.e., they require tuning appropriate situation-specific parameters).

Finally, [47] approach the incentive problem (finding an efficient way to incentivise participants so as to maximise the overall utility of the requesters) by combining the classical principal-agent model and MABs. In particular, they formalise the incentive problem as a multiple-round process. In each round, one participant (i.e., worker) completes a task based on a contract designed in advance by the requester. From the performance of the participants so far, their algorithm (AgnosticZooming) helps the requester adaptively adjust the contracts to be used in a round so that the requester's utility is maximised. Each potential contract is treated as an arm in their algorithm. However, they only consider financial incentives in the form of monotone contracts where the outcomes are not lower with higher payments. This prevents the algorithm from being used effectively in crowdsourcing projects where the motivation for participation is not only money, but can be human capital advancement or community identification [48]. In these projects, the outcomes might not be proportional to payments [27, 43]. This might affect the performance of their algorithm in crowdsourcing projects whose time budgets are critical, as it takes a long time to identify a good contract.

Our preliminary work on the ISP [20] focuses on the case where candidate incentives are many (compared to the budget) and there exist correlations between the incentives. Specifically, as presented in Section 1, some candidate incentives might have the same parameters but the values of the parameters are different, so their performance is likely to be correlated. In fact, these incentives are chosen from the same *incentive method* (thus they have the same parameters) but are different in the way the method is implemented (i.e., different values of the parameters). Here, an incentive method can be any incentive in the literature, such as paying for performance, using bonuses, or using contests. Also, *correlations* between incentives means that the difference in the effectiveness between two adjacent incentives (i.e., the values of their parameters are slightly different) is assumed to be small. Indeed, it is likely that when the parameter values of an incentive method change slightly, the effectiveness of the corresponding incentives also changes gradually [42, 49].

In the paper, correlated incentives are grouped into *clusters*. Also, the proposed model is for the ISP where we have no or very little prior knowledge about the performance of participants in the project of interest and the budget for the project is large compared to the chosen incentives.

Although the algorithm proposed (BOIS) is shown to solve the ISP effectively, in reality, many crowdsourcing projects do not have such large budgets. Also, in some projects, we might have good prior knowledge about participant performance, so in each cluster, we can continue choosing some candidate incentives with a high confidence that they are good. In some other projects, the budgets are not large enough (compared to the chosen candidate incentives).

Thus, we need to continue choosing some incentives in each cluster so that after exploring the incentives, we have sufficient budget to exploit the best incentive explored. BOIS is not effective in these projects, since the incentives are not correlated. This is because BOIS focuses more on the correlations to quickly identify the best incentive in each cluster. So, when there are only a small number of candidate incentives in each cluster, the algorithm does not have a good exploration-exploitation balance compared to the algorithm proposed in this paper (HAIS). Generally, in these projects, BOIS is somewhat similar to Stepped fKUBE.[5] More specifically, Stepped fKUBE spends the first period obtaining initial estimates of the incentives. Then, it spreads a certain portion (specified by ϵ_2) of the residual budget across the next periods (except the last one) to apply the best incentive so far (identified by their upper confidence bounds). After that, in the last period, it simply applies the best incentive with the remaining budget. As will be shown in Section 6.3, HAIS outperforms Stepped fKUBE in these projects.

3 The incentive selection problem

In this section, we first describe the incentive selection problem (ISP). Then, we formalise it as a batched 2d-budgeted group-based MAB problem.

Suppose a requester wants to run a crowdsourcing project. The objective is typically to maximise the requester's overall utility with a given budget before a given time. We can include task quantity, task quality, task completion time, or some subset of them in the utility function.[6] For

---

[5]The Stepped fKUBE algorithm will be presented in detail in Section 6.1.

[6]A note when choosing an aspect to be included in the metric is that it should not only measure the effectiveness of contest implementations (i.e., incentives) appropriately, but it should also be possible to evaluate the effectiveness easily and ideally in an automatic manner. That is because after deploying a contest of a specific implementation in a period of time, the effectiveness of this implementation should be calculated quickly before deploying another contest (to another group of participants) in the next period. One example of such tasks is drawing an evacuation route from a building to the nearest road as described in [29] and the corresponding metric is the number of valid evacuation routes completed. A valid evacuation route can be simply defined as the one that really connects a building to a road. This can be done automatically by checking if the route starts somewhere inside the outline of the building and ends at a road. We can manage a higher level of quality by defining a valid evacuation route as one that really connects a building through an entrance to a road with a walkway or an open space. However, to do this, we have to spend more time on manually validating submitted routes. We can do this by, for example, majority voting. In detail, we can ask some other participants to check if a route is valid or not and then choose the decision from the majority. This can be done in the form of another task. So, the total time will be expanded significantly.

example, [25] consider the quantity and quality of the tasks. To achieve this objective, we spend the available budget on providing incentives to encourage participants (referred to as *users*) to perform tasks. The incentives can be in the form of contests or individual-based (i.e., non-contests). They can be different in terms of the number of users in a contest (note that with individual-based incentives, this value is always one), the performance evaluation method, or the prize distribution. Since the effectiveness of incentives is usually unknown in advance, we are interested in finding an efficient means of selecting candidate incentives (i.e., exploring their effectiveness and then exploiting the most effective one) to maximise the requester's utility. We refer to this as the *incentive selection problem* (ISP).

Formally, let $\mathbf{I} = \{1, 2, \ldots, I\}$ denote a set of incentives that are being considered for use in a crowdsourcing project. Each incentive has a group size (the number of users in a group that is offered this incentive) and a cost (of offering the incentive to a group of users). The cost of each incentive is deterministic. For example, there may be 3 incentives. Incentive 1 can be contests of five users, where the base payment is £0.50 and the prize for the best user (who performs the most tasks in a contest) is £2.50. That means this incentive has 5 as the group size and £5.00 as the cost (£2.50 for the base payments and £2.50 for the prize). Incentive 2 is almost the same as incentive 1, but the base payment and the prize for the best user are £0.70 and £1.50. Similarly, incentive 3 can also be contests, whereby there are ten users in a contest, the base payment is £0.50, the prize for the best user is £1.50, and the prize for the second best one is £1.00.

The number of incentives (I) also corresponds to the number of arms in a MAB problem. Pulling arm i corresponds to offering incentive i to a group of g_i (referred to as *group size*) users in a specific *time period* (or *period* for short, e.g., five hours or one day). The periods do not overlap and are denoted by $t = 1, 2, \ldots$. Each incentive can be applied to different groups in the same or different periods. We can only start period t (i.e., applying incentives to other groups in period t) when all groups in period $t - 1$ are finished. To illustrate this, Fig. 1 shows an example where three incentives are applied to various groups over five periods (corresponding to days here). On the first day ($t = 1$), we apply incentive 1 to four groups, incentive 2 to two groups, and incentive 3 to three groups. Here, the group sizes of incentives 1, 2, and 3 are 2, 4, and 4 respectively.

Let $\mathbf{N} = \left\{ n_i^{(t)} \mid t = 1, 2, \ldots; \ i = 1, \ldots, I \right\}$ denote a policy of applying the incentives (or *applying policy* for short), where $n_i^{(t)}$ is the number of times incentive i is applied in period t (i.e., incentive i is offered to $n_i^{(t)}$ different groups). For instance, in the example shown in Fig. 1, $n_1^{(1)} = 4$, $n_2^{(1)} = 2$, and $n_3^{(1)} = 3$. Applying incentive i incurs a fixed

Fig. 1 An example of an applying policy where $I = 3$, $g_1 = 2$, $g_2 = 4$, $g_3 = 4$, and $T = 5$ (days)
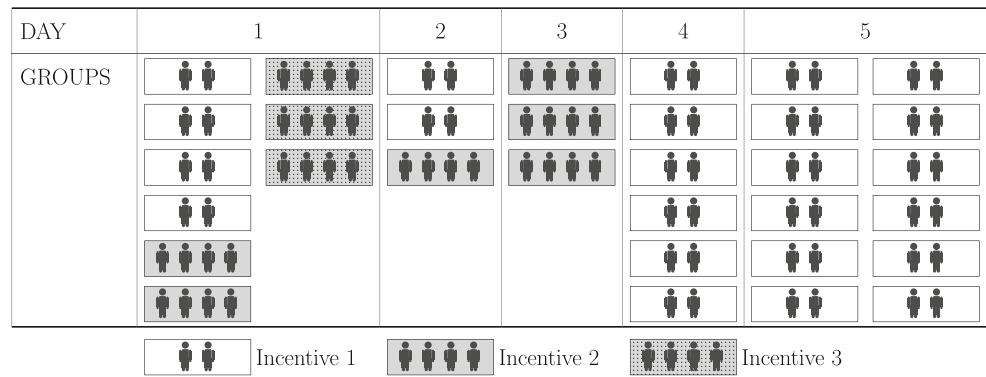
cost of c_i and enjoys a *utility* which is drawn independently from a fixed unknown distribution with an unknown mean (i.e., expectation or expected value) μ_i. Let $r_i^{(t)}$ be the total utility of applying incentive i $n_i^{(t)}$ times in period t. Note that $r_i^{(t)}$ is the total utility of users in all groups of incentive i in period t. The objective is to find an applying policy that maximises the expectation of the overall utility with a given financial budget B and time budget T:

$$\max \sum_{t=1}^{T} \sum_{i=1}^{I} n_i^{(t)} \mu_i \quad \text{subject to} \quad \sum_{t=1}^{T} \sum_{i=1}^{I} n_i^{(t)} c_i \leq B.$$

From the definition above, we can see that the ISP is a batched 2d-budgeted group-based MAB. Indeed, in each period, each arm can be pulled several times (i.e., an incentive can be offered to different groups) and multiple arms can be pulled (i.e., several incentives can be offered). Hence, there is *batched* in the name. Also, the pullings are constrained by a financial budget B (a dimension) and a time budget T (another dimension). Thus, the problem is *2d-budgeted*, where "d" means "dimension". Moreover, one characteristic that makes the ISP different from other MAB models studied in the literature is the *group-based* nature of the arms. For ease of presentation, in some places, when it does not lead to confusion between the two types of budget, we use *budget* for the financial one.

Also in this work, we only consider the contests where the submissions of all users (not only the best user) in a contest are useful to the requester. In other words, the utility is additive. This assumption prevents the model being applied in projects where the requesters only consider the best submission in every contest. Yet, this normally happens in design contests. For example in crowdsourcing systems for design tasks such as 99 Designs (99designs.com), Design Crowd (www.designcrowd.com), and Crowd Spring (www.crowdspring.com), only the best submissions are used, the other submissions are discarded. However in microtask crowdsourcing projects, for example, every task completed is typically useful to the requesters [29, 50].

Thus, although the assumption may limit the applications of the model, it is reasonable in many crowdsourcing projects.

4 The `HAIS` algorithm

Here we introduce `Hoeffding-based Adaptive Incentive Selection` (henceforth, `HAIS`), an adaptive algorithm for the ISP. `HAIS` uses several heuristics[7] (as will be presented in Section 4.5) to help solve the ISP effectively.

However, we first detail how the algorithm and the benchmarks measure the effectiveness of the incentives (Section 4.1). We then briefly present Hoeffding's inequality and discuss how we will utilise this inequality when dealing with the ISP (Section 4.2). After that, we give an overview of the algorithm (Subsection 4.3). Finally, we detail how `HAIS` is built and how it acts in the exploration (Sections 4.4 and 4.5) and exploitation (Sections 4.6 and 4.7) phases.

4.1 Measuring the effectiveness of the incentives

To measure the effectiveness of the incentives, we use *density* (i.e., the utility-cost ratio) [30], as it reflects the average utility (i.e., reward in the context of MABs) obtained per cost unit. The density of incentive i is defined as $\delta_i = \mu_i / c_i$, where c_i is the cost of applying the incentive once and μ_i is the mean utility. However, as the real densities of the incentives are unknown a priori, we have to

---

[7] `HAIS` is not a metaheuristics algorithm, since it is designed to deal with the ISP specifically, while a metaheuristics algorithm should provide a generic framework to solve many different problems such as simulated annealing, tabu search, or genetic algorithms [51]. `HAIS` has exploration and exploitation concepts which are similar to diversification and intensification as in metaheuristics algorithms. However, the exploration-exploitation trade-off in `HAIS` is inherently from the MAB problem itself, which is to identify the best arms. This is different from the diversification-intensification balance in metaheuristics, which is to find the best solution in the combinatorial search space.

estimate them. Right after period t, the estimate of incentive i's density is:

$$d_i^{(t)} = \hat{\mu}_i^{(t)}/c_i, \tag{1}$$

where $\hat{\mu}_i^{(t)} = (1/m_i^{(t)}) \sum_{j=1}^{t} r_i^{(j)}$ is the current estimate of incentive i's mean utility ($m_i^{(t)} = \sum_{j=1}^{t} n_i^{(j)}$ is the number of times incentive i has been applied until the end of period t). With all algorithms examined in this work, each arm will be pulled at least once in period 1 and the estimates will be conducted from period 2. So, in (1), $m_i^{(t)} > 0 \,\forall i = 1, \ldots, I$.

To keep the presentation simple, we use *the best (worst) incentive* to denote the incentive with the highest (lowest) estimate, as opposed to *the real best (worst) incentive*. Also, we use *the estimate of an incentive* instead of *the estimate of an incentive's density (or effectiveness)*.

4.2 The Hoeffding's inequality

In general, this inequality is used to determine the number of samples needed to obtain a certain level of confidence for a confidence interval around the expected value of

Fig. 2 An example of running HAIS where $I = 3$, $g_1 = 4$, $g_2 = 2$, $g_3 = 2$, $T = 5$ (days), $B = £80$, $c_1 = £4$, $c_2 = £2$, $c_3 = £2$, $\epsilon_1 = 0.4$, $\epsilon_2 = 0.5$, $U_1 = 8$, incentive 2 is the real best incentive, incentive 3 is the real worst
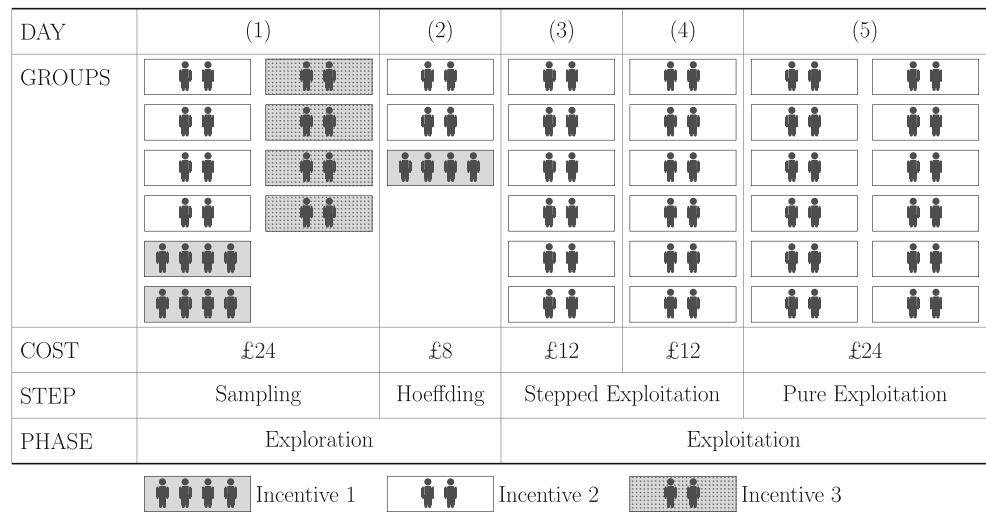
(a) A view about a pulling strategy over 5 days
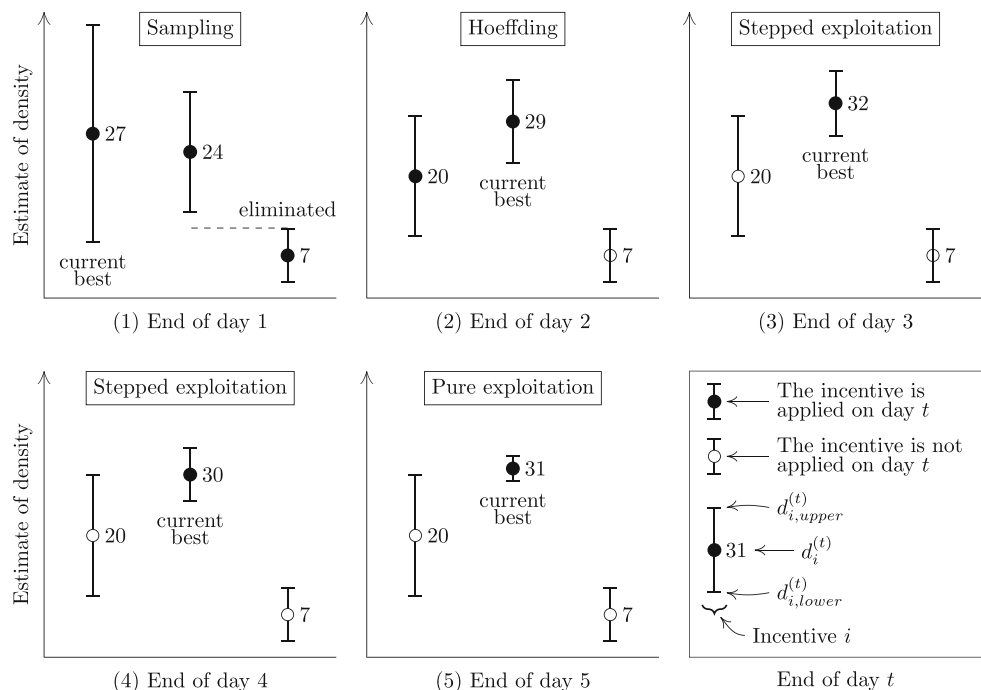
(b) A view about the estimates of the incentives over 5 days

the samples. In particular, let $Y_1, \ldots, Y_n$ be independent random variables with $Y_i \in [y_{min}, y_{max}]$ for all i, where $-\infty < y_{min} \leq y_{max} < +\infty$. Then, Hoeffding's inequality [52] states that:

$$P(\bar{Y} - E[\bar{Y}] \geq \gamma) \leq \exp\left(\frac{-2n\gamma^2}{(y_{max} - y_{min})^2}\right) \text{ and} \quad (2)$$

$$P(\bar{Y} - E[\bar{Y}] \leq -\gamma) \leq \exp\left(\frac{-2n\gamma^2}{(y_{max} - y_{min})^2}\right), \quad (3)$$

for all $\gamma \geq 0$, where $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$ and $E[\bar{Y}]$ is the expected value of $\bar{Y}$ (Fig. 2).

Applying this to the ISP, we can determine the number of sampled users needed in particular incentives to obtain good estimates of the incentives. And hence, we can identify the best incentive with high enough confidence, i.e., greater than or equal to a certain level of confidence which is specified in advance (e.g., 50%). We call this confidence level L_h (h for Hoeffding). We choose Hoeffding's inequality as it helps us identify the appropriate numbers of times the incentives should be applied in the exploration phase dynamically based on the estimates of the incentives so far. Concretely, the inequality is applied to determine the number of additional users needed on each incentive and then based on the group size of the incentive to identify the number of times the corresponding incentive is applied.

For example, in a crowdsourcing project, there are two incentives which have the group sizes of 10 and 5 respectively. And suppose the chosen value of L_h is 50% (i.e., the confidence level of identifying the best incentive after applying the incentives so that each incentive has the target number of sampled users is 50%). In order to obtain initial estimates, the target number of sampled users in each incentive is at least 30 (a parameter which is chosen in advance[8]). Thus, in the first period, incentive 1 is applied three times (to have 3 * 10 = 30 sampled users) and incentive 2 is applied six times (to have 6 * 5 = 30 sampled users). Then, after applying Hoeffding's inequality, suppose the result suggests that to obtain a confidence level of $L_h = 50\%$ in being sure that the current best incentive is the real best one, each incentive needs to have at least 60 sampled users.[9] Since currently incentive 1 (with group size of 10) already has 30 sampled users, it needs 30 more. That means we need to apply this incentive three more times. Similarly, as incentive 2 (with group size of 5) already has 30 sampled users, we need to apply this incentive six times to have 30 more.

In terms of the value of L_h, it should be chosen to be high enough (e.g., 50%, rather than only 10%), so that the current best incentive is likely to be a highly effective incentive. However, it should not be too high, as the algorithm might spend the budget on applying less effective incentives. The advantage of using the predefined parameter L_h is that we could choose a fixed value for it (e.g., 50%) in all crowdsourcing projects. With each project, based on L_h and the estimates of the incentives so far, HAIS will adaptively identify an appropriate number of sampled users needed on each incentive.

4.3 Algorithm overview

HAIS splits the application of the incentives into two phases: exploration and exploitation. In the first phase, it has two steps: sampling and Hoeffding. In the second phase, it also has two steps: stepped exploitation and pure exploitation. We next provide an overview of HAIS over the four steps following an illustrative example of how the algorithm works over these steps. The diagram in Fig. 3 shows the connections between the steps in the full process of the algorithm.

The *sampling* step is conducted in the first period. The purpose of this step is to obtain initial estimates of the incentives in order to apply Hoeffding's inequality in the next period. Specifically, in this step, HAIS applies the incentives so that each incentive has a minimum number of sampled users. This number, which is referred to as U_1, is specified in advance. U_1 should be large enough (e.g., 20) to obtain significant estimates of the incentives. Yet, it should not be too large to take up a large portion of the budget, e.g., 200. After that, the algorithm eliminates clearly ineffective incentives by comparing the confidence intervals of the estimates. Concretely, an incentive i will be eliminated if there exists another incentive j whose lower bound of the confidence interval is larger than the upper bound of the confidence interval of this incentive (i.e., $d_{i,upper}^{(1)} < d_{j,lower}^{(1)}$). Eliminated incentives will not be applied in the Hoeffding step. Beside the estimates of the incentives, to calculate the corresponding confidence intervals, we need to set a value for the level of confidence. This value is a predefined parameter for the algorithm which is referred to as L_e (e means elimination) [10], e.g., 95%.

In the second period, HAIS applies *Hoeffding*'s inequality as described in Section 4.2 to have better estimates of the incentives, preparing for exploitation after that. One issue that might occur in the exploration phase is that, as the performance of users in each incentive is stochastic, the number of sampled users suggested by Hoeffding's inequality can

---

[8]This parameter (which is called U_1) will be presented in detail in Section 4.3.

[9]This number (which is called U_2, shown in (15)) is the result of applying Hoeffding's inequality, which will be presented in detail in Section 4.5.

[10] HAIS uses eight predefined parameters which are shown in Table 2. Section 6.2.2 presents how HAIS chooses values for these parameters.

Fig. 3 High-level overview of the HAIS algorithm. See Algorithm 2 for details of the input, output, predefined parameters, steps 1–4, and decisions a–b. "Time is up" in (b) means all provided (T_s) periods for stepped exploitation have been used
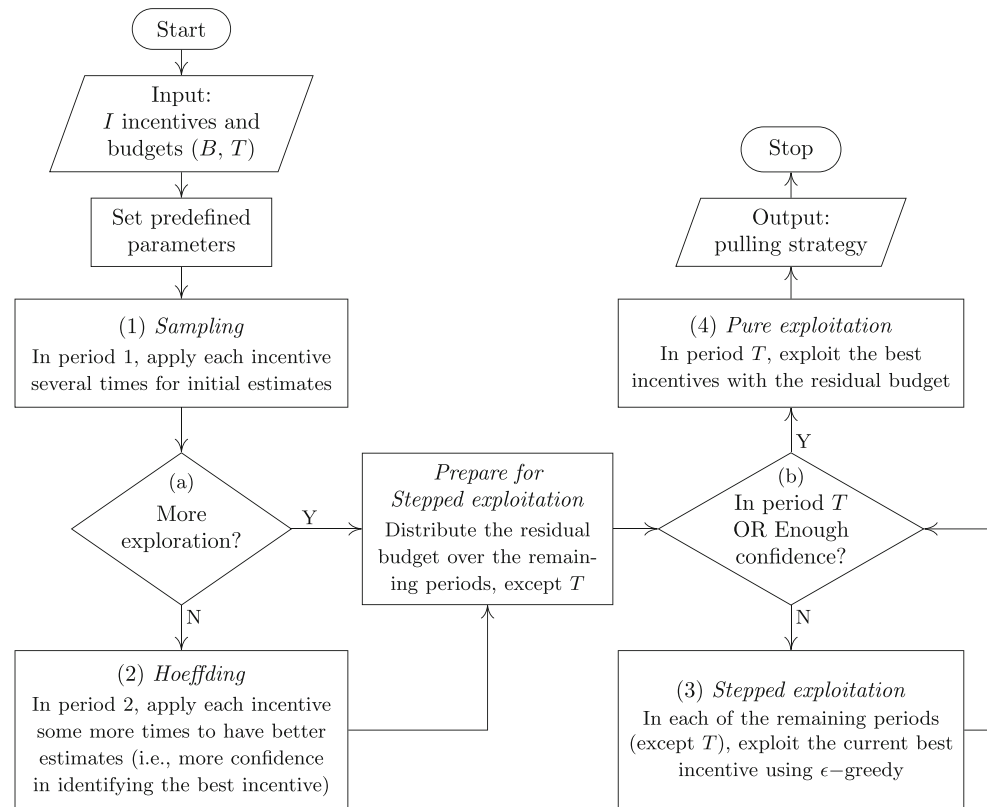
be very large in some cases and this might use up a large portion of B. This is ineffective since although it better estimates the incentives, it does not have much budget left to exploit the best incentives explored. Thus, we adapt the idea from ϵ-first of using a limited financial budget for exploration (specified by a predefined parameter which is referred to as $\epsilon_1 \in (0, 1)$). This budget bound for exploration (i.e., $\epsilon_1 B$) is applied to both sampling and Hoeffding steps. Although both HAIS and ϵ-first use the same parameter ϵ_1, they have different purposes. In ϵ-first, ϵ_1 is used to identify the budget for exploration. So, it should be changed appropriately based on specific situations. In particular, with ϵ-first, in projects where the financial budget is large, we should choose small values of ϵ_1, such as 0.02, to prevent spending a large proportion of the budget on exploring the incentives. And in projects where the financial budgets are small, we should choose large values of ϵ_1, such as 0.1, to have sufficient budget to explore. In contrast to this, HAIS uses L_h as the main parameter to control the budget for exploration. L_h (as mentioned in Sub-Section 4.2), is the level of confidence to identify the best incentive. HAIS only uses ϵ_1 as an upper bound for the budget for exploration. Hence, the parameter ϵ_1 in HAIS can be chosen intuitively, such as 0.1, and it does not have to be changed in different projects.

In the next periods (except the last one), it conducts *stepped exploitation*, which takes advantage of the remaining periods to exploit effectively. More specifically, it splits the residual budget (b) into two parts based on a predefined $\epsilon_2 \in (0, 1)$ (e.g., 0.5 to have two equal parts) [10]. Then, it distributes the first part, $\epsilon_2 b$, equally across T_s periods, where $T_s + 1$ is the remaining periods including the last one.

In each of these T_s periods, HAIS applies ϵ-greedy [53] with the given budget ($\epsilon_2 b/T_s$). Specifically, with a predefined $\epsilon_{greedy} \in [0, 1)$, it applies a random incentive with probability ϵ_{greedy} and the current best incentive with probability $1 - \epsilon_{greedy}$, followed by an update to the estimate of the incentive applied. Here, the parameter ϵ_{greedy} is to control the level of exploration in this step. The reason for using ϵ-greedy in this step is that, although the algorithm is focusing on exploiting the best incentives explored so far, it can continue doing some more exploration with other incentives. This prevents ignoring the best incentive which has a low estimate after the Hoeffding step. The second part, $(1 - \epsilon_2)b$, is spent in the last period to *purely exploit* the best incentives, that is to apply the best incentives with the residual budget. Figure 3 shows a high-level overview of HAIS, which shows the above-mentioned four steps in the whole process of the algorithm. Also, a simplified version of the algorithm is presented in Algorithm 1.

Algorithm 1 Simplified version of the `HAIS` algorithm.

Input: financial budget (B), time budget (T), number of incentives (I), $\cdots$
Predefined parameters: ϵ_1, ϵ_2, U_1, L_h, $\cdots$
Output: applying policy, overall utility, number of periods used
Note: See Algorithm 2 for the fully-detailed version of the `HAIS` algorithm.

Sampling {
01: Apply all incentives so that each incentive has about U_1 sampled users.
02: Update the estimates of all incentives, i.e., d_i, $d_{i,lower}^{(1)}$ and $d_{i,upper}^{(1)}$ $\forall i = 1 \ldots I$.
03: Eliminate clearly ineffective incentives from being applied in the Hoeffding step. An incentive (i) is eliminated if there exists another incentive (j) whose the lower bound of the estimate ($d_{j,lower}^{(1)}$) is larger than the upper bound of its estimate ($d_{i,upper}^{(1)}$).

Hoeffding {
04: Calculate the target number of sampled users (U_2) so as to identify the best incentive with a confidence level of L_h (see Section 4.2).
05: Apply all active incentives so that each incentive has about U_2 sampled users.
06: Update the estimates of all active incentives.

Stepped Expl. {
07: The budget for stepped exploitation is $b_2 = \epsilon_2 b$, where b is the residual budget.
08: This budget (b_2) is distributed across T_s periods, where $T_s + 1$ is the number of remaining periods including the last one.
09: In each of these T_s periods, with the given budget, apply the current best incentive with probability $1 - \epsilon_{greedy}$ and a random incentive with probability ϵ_{greedy}; then update the estimate of the incentive just applied.
10: *Note that this step might not use all T_s periods* (see Section 4.6).

Pure Expl. {
11: Apply the best incentive with the residual budget.
12: Apply the second best incentive with the residual budget.
13: Repeat the process until the residual budget is not enough to apply any incentive.

14: **return** the applying policy, the overall utility, and the number of periods used.

To illustrate the algorithm, Fig. 2 shows an example of how `HAIS` acts in a simple case. In the first period of the example (day 1), incentives 2 and 3 are applied four times, while incentive 1 is applied only twice, to have enough $U_1 = 8$ users (Fig. 2a$^{(1)}$). Note that the numbers chosen in this example (e.g., U_1 or g_i) are for illustrative purposes only. After this period, the estimate of incentive 3 is significantly lower than that of incentive 1, i.e., $d_{3,upper}^{(1)} < d_{2,lower}^{(1)}$ (Fig. 2b$^{(1)}$). Incentive 3 is therefore eliminated. Hence, in the Hoeffding step conducted in period 2, `HAIS` decides to apply incentives 1 and 2, so that it has an additional 4 users for each incentive with an expectation to differentiate the incentives' effectiveness with at least $L_h = 50\%$ confidence (Fig. 2a$^{(2)}$). After the exploration phase, the estimate of incentive 2 (29) is higher than that of incentive 1 (20) (Fig. 2b$^{(2)}$). Thus, incentive 2 is applied in the third period (day 3), followed by an update to this incentive's estimate. Note that, on day 3, incentives 1 and 3 were not applied (Fig. 2b$^{(3)}$). In the next period (day 4), as the estimate of incentive 1 (30) still appears to be the highest, `HAIS` just applies this incentive with the given budget (£12). In the last period, it applies the best incentive (incentive 2) 12 times with the remaining budget £24 (Fig. 2a$^{(5)}$ and Fig. 2b$^{(5)}$).

To summarise, the key novelty of `HAIS` is that it combines three techniques that together result in an adaptive and efficient way to balance exploration and exploitation.

First, it uses Hoeffding's inequality to identify how much exploration is sufficient to find the real best incentive with a certain level of confidence. This allows `HAIS` to adaptively distribute the budget for exploration without tuning any situation-specific parameters.

Second, the algorithm applies each incentive several times in the first round to obtain initial estimates of the densities of the incentives, together with using confidence intervals to eliminate clearly ineffective incentives after this period.

Third, it makes use of the time budget to continue exploring while exploiting the incentives by spreading the residual budget across the remaining periods.

In the following subsections, details of the four steps will be discussed. The explanations will be linked to the corresponding parts of the pseudocode of `HAIS` shown in Algorithm 2.

Algorithm 2 The `HAIS` Algorithm.

Input:
  $B, T, I,$ ▷ financial budget, time budget, number of incentives
  $g_i, c_i \ (\forall i = 1, \dots, I)$ ▷ group sizes and costs of the incentives

Predefined parameters:
  $\epsilon_1, \epsilon_2, \epsilon_{greedy}, U_1, L_e, L_h, L_s, N_s$ ▷ see Table 2 for their descriptions

Output:
  $r, t,$ ▷ overall utility, number of periods used
  $\mathbf{N} = \left\{ n_i^{(t)} \mid t = 1, \dots, T; \ i = 1, \dots, I \right\}$ ▷ applying policy

Note: `ApplyIncentive`(i, n) is to apply incentive i n times and return the total utility.

01: $b \leftarrow B; n_i^{(t)} \leftarrow 0 \ \forall t = 1, \dots, T; \ i = 1, \dots, I;$
        ▷ overall residual budget; init # of times each incentive is applied in each period

Sampling {

02: $t \leftarrow 1; b_1 \leftarrow \epsilon_1 B;$ ▷ start the first period; residual budget for exploration

03: $u_1 \leftarrow \min \left\{ U_1, \frac{b_1}{\sum_{i=1}^{I} c_i/g_i} \right\};$ ▷ target # of sampled users on each incentive after period 1

04: **for** $i = 1 \rightarrow I$ **do**

05:     $n_i^{(t)} \leftarrow [u_1/g_i]; r_i^{(t)} \leftarrow$ `ApplyIncentive`$(i, n_i^{(t)});$

06:     $b \leftarrow b - c_i n_i^{(t)}; b_1 \leftarrow b_1 - c_i n_i^{(t)};$ Update $d_i^{(t)}$ using Equation 1;

07:     Update $d_{i,lower}^{(1)}$ and $d_{i,upper}^{(1)}$ using Eq. 4; ▷ conf. interval of incentive i's estimate

08: $\mathbf{A} \leftarrow \left\{ i \mid i \in \{1, \dots, I\}, \forall j \neq i : d_{j,lower}^{(1)} \leq d_{i,upper}^{(1)} \right\};$ ▷ set of active incentives

}

Hoeffding {

09: **if** $(b_1 \geq \sum_{i \in \mathbf{A}} c_i)$ and $(|\mathbf{A}| > 1)$ **then**   ▷ enough budget and >1 active incentive

10:     Calculate U_2 using Equation 15; $u_2 \leftarrow \min \left\{ U_2, \frac{b_1}{\sum_{i \in \mathbf{A}} c_i/g_i} + u_1 \right\};$

11:     **if** $u_2 > u_1$ **then**   ▷ check if further exploration is needed
12:         $t \leftarrow 2;$ ▷ start period 2
13:         **for** $i \in \mathbf{A}$ **do**   ▷ pull the active incentives

14:             $n_i^{(t)} \leftarrow \left[(u_2 - n_i^{(1)} * g_i)/g_i \right]; r_i^{(t)} \leftarrow$ `ApplyIncentive`$(i, n_i^{(t)});$

15:             $b \leftarrow b - c_i n_i^{(t)}; b_1 \leftarrow b_1 - c_i n_i^{(t)};$ Update d_i using Equation 1;

16: $\mathbf{A} \leftarrow \{1, \dots, I\};$ ▷ all incentives are now active

}

Stepped Exploitation {

17: Calculate confidence level $l^{(t)}$ using Equation 17;

18: $b_2 \leftarrow \epsilon_2 b;$ ▷ residual budget for stepped exploitation

19: **while** $(t < T - 1)$ and $(b_2 \geq \min_{i \in \{1, \dots, I\}} c_i)$ and $(l^{(t)} < L_s)$ and $(ns_i^{(t)} < N_s)$ **do**

20:     $t \leftarrow t + 1; i \leftarrow \begin{cases} \operatorname{argmax}_{i \in \{1, \dots, I\} \mid c_i \leq b} \{ d_i \} & \text{with probability } 1 - \epsilon_{greedy} \\ \operatorname{random} \{ i \in \{1, \dots, I\} \mid c_i \leq b \} & \text{with probability } \epsilon_{greedy} \end{cases};$

21:     $n_i^{(t)} \leftarrow \max\{ 1, [b_2/(c_i T_s)] \}; r_i^{(t)} \leftarrow$ `ApplyIncentive`$(i, n_i^{(t)});$

22:     $b \leftarrow b - c_i n_i^{(t)}; b_2 \leftarrow b_2 - c_i n_i^{(t)};$ Update d_i using Equation 1;

23:     Calculate confidence level $l^{(t)}$ using Eq. 17; ▷ to consider stop stepped exploiting

}

Pure Exploit. {

24: $t \leftarrow t + 1;$

25: **while** $b \geq \min_{i \in \{1, \dots, I\}} c_i$ **do**

26:     $i \leftarrow \operatorname{argmax}_{i \in \{1, \dots, I\} \mid c_i \leq b} \{ d_i \};$

27:     $n_i^{(t)} \leftarrow \lfloor b/c_i \rfloor; r_i^{(t)} \leftarrow$ `ApplyIncentive`$(i, n_i^{(t)}); b \leftarrow b - c_i n_i^{(t)};$

}

28: $r \leftarrow \sum_{t=1}^{T} \sum_{i=1}^{I} r_i^{(t)};$

29: **return** $r, t, \mathbf{N};$

4.4 The sampling step

As discussed above, the objective of this step is twofold: to obtain initial estimates of the incentives and to preclude clearly ineffective incentives from being used in the next step (Hoeffding). Regarding the implementation of this step, it first determines a target number of users that should be sampled on each incentive after this step (i.e., after the first period), u_1 (Line 3). If the budget is large enough, this number can be set to U_1 (the expected number of sampled

users in the sampling step). However, as discussed in the previous subsection, when the budget to have U_1 users sampled on each incentive exceeds the maximum budget for exploration $\epsilon_1 B$, u_1 will be set to a smaller value so that the budget to have u_1 users sampled on each incentive is about $\epsilon_1 B$. If this happens, the Hoeffding step will be skipped as the budget for exploration is exceeded. Since the group sizes of the incentives are different, we approximate the limited number of users corresponding to this budget bound by dividing $\epsilon_1 B$ by the total cost of one user on each incentive, which is $\sum_{i=1}^{I} c_i / g_i$ (g_i is the group size of incentive i). The purpose of the budget bound for exploration is to prevent spending too much budget on exploration. So, with this purpose in mind, the actual cost for exploring does not need to be strictly within the bound. This means it can be slightly more than this number. Given this, the above-mentioned approximation is acceptable.

Based on the target number of users u_1 and the group size of each incentive g_i, the number of times each incentive should be applied is calculated by rounding the division u_1/g_i to the nearest integer (Line 5). Then the incentive is applied (Line 5) followed by an update on the estimate of this incentive (Line 6) and an update on the confidence interval of the incentive's estimate (Line 7). The confidence interval of incentive i's estimate $\left(d_{i,lower}^{(1)}, d_{i,upper}^{(1)}\right)$ is:

$$d_i^{(1)} \pm z_e \frac{s_i^{(1)}}{\sqrt{n_i^{*(1)}}}. \tag{4}$$

In this equation,

- $t = 1$ as the calculation is at the end of the first period;
- z_e is the critical value (z-value) corresponding to the confidence level L_e;
- $n_i^{*(1)} = n_i^{(1)} g_i$ is the number of sampled users of incentive i at the end of the first period;
- $s_i^{(1)} = \frac{1}{c_i^*}\sqrt{\frac{\sum_{u=1}^{n_i^{*(1)}} \left(r_{i,u}^{(1)} - \bar{r}_i^{(1)}\right)^2}{n_i^{*(1)} - 1}}$ is the estimate of the standard deviation of incentive i's density at the end of period 1; where $c_i^* = c_i / g_i$ is the average cost of a user in incentive i, $r_{i,u}^{(1)}$ is the utility created by the uth user in incentive i in period 1, and $\bar{r}_i^{(1)} = \sum_{u=1}^{n_i^{*(1)}} r_{i,u}^{(1)} / n_i^{*(1)}$ is the average of the utility received from all users in incentive i at the end of period 1.

Finally, based on the confidence intervals of the estimates, HAIS determines the set of incentives to be applied in the Hoeffding step, $\mathbf{A}$ (Line 8). The incentives that belong to $\mathbf{A}$ are referred to as *active incentives*. The others are eliminated and will not be applied in the Hoeffding step. Although the eliminated incentives will not be applied in the Hoeffding step, these incentives can be

applied afterwards (Line 16). This helps us ensure we do not miss the real best incentive which is eliminated in the first period because of a low estimate compared to other incentives.

4.5 The Hoeffding step

We now describe how HAIS uses Hoeffding's inequality to calculate the number of times each active incentive should be applied in the subsequent period so that a level of confidence of at least L_h can be obtained in identifying the real best incentive.

Let $X_{i,1}^{(t)}, \ldots, X_{i,u_i^{(t)}}^{(t)}$ denote the utility per cost unit of $u_i^{(t)}$ sampled users in incentive i from the beginning until the end of period t. The second value of each subscript denotes a specific sampled user. For example, $X_{i,5}^{(t)}$ is the utility per cost unit of the 5th sampled user. They can be considered as $u_i^{(t)}$ random variables whose values are bounded in $[\beta_i^{min}, \beta_i^{max}]$. According to Hoeffding's inequality, we have:

$$P(\bar{X}_i^{(t)} - \delta_i \geq \gamma) \leq \exp\left(-2u_i^{(t)}\gamma^2/\beta_i^2\right) \text{ and} \tag{5}$$

$$P(\bar{X}_i^{(t)} - \delta_i \leq -\gamma) \leq \exp\left(-2u_i^{(t)}\gamma^2/\beta_i^2\right), \tag{6}$$

where $\gamma > 0$, $\beta_i = \beta_i^{max} - \beta_i^{min}$, and $\bar{X}_i^{(t)} = \frac{1}{u_i^{(t)}} \sum_{u=1}^{u_i^{(t)}} X_{i,u}^{(t)}$.

From (5) and (6), we have:

$$P(\bar{X}_i^{(t)} - \delta_i < \gamma) \geq 1 - \exp\left(-2u_i^{(t)}\gamma^2/\beta_i^2\right) \text{ and} \tag{7}$$

$$P(\bar{X}_i^{(t)} - \delta_i > -\gamma) \geq 1 - \exp\left(-2u_i^{(t)}\gamma^2/\beta_i^2\right). \tag{8}$$

Applying (7) to the worst (active) incentive i_1 [11], the resulting confidence level that $\bar{X}_{i_1}^{(t)} - \delta_{i_1} < \gamma_{i_1}$ is $l_1^{(t)} = 1 - \exp\left(-2u_{i_1}^{(t)}\gamma_{i_1}^2/\beta_{i_1}^2\right)$, or:

$$\gamma_{i_1} = \beta_{i_1}\sqrt{\frac{\ln\left(1/(1 - l_1^{(t)})\right)}{2u_{i_1}^{(t)}}}. \tag{9}$$

Similarly, applying (8) to the best incentive i_2 [11], the resulting confidence level that $\bar{X}_{i_2}^{(t)} - \delta_{i_2} > -\gamma_{i_2}$ is:

$$\gamma_{i_2} = \beta_{i_2}\sqrt{\frac{\ln\left(1/(1 - l_2^{(t)})\right)}{2u_{i_2}^{(t)}}}. \tag{10}$$

To differentiate the effectiveness of the two incentives, the confidence intervals γ_{i_1} and γ_{i_2} must be small enough

---

[11] To keep the presentation simple, we use i_1 and i_2 to denote the worst and best incentives respectively at the end of period t instead of $i_1^{(t)}$ and $i_2^{(t)}$. These incentives are identified by $i_1 \stackrel{\text{def}}{=} i_1^{(t)} = \arg\min_{i \in \mathbf{A}: c_i \leq b^{(t)}} \left\{d_i^{(t)}\right\}$ and $i_2 \stackrel{\text{def}}{=} i_2^{(t)} = \arg\max_{i \in \mathbf{A}: c_i \leq b^{(t)}} \left\{d_i^{(t)}\right\}$, where $b^{(t)}$ is the residual budget after finishing period t.

compared to the distance between the expected values of the two incentives' densities:

$$\gamma_{i_1} + \gamma_{i_2} \leq \delta_{i_2} - \delta_{i_1}. \tag{11}$$

This is illustrated in Fig. 4. The intuition about finding the real best incentive by comparing the best and worst incentives is that the purpose of the exploration phase is to quickly identify an incentive which has a high density (compared to others), not the real best incentive. Then, in the exploitation phase, the algorithm can gradually find the real best incentive with higher confidence by continuously updating the incentives' estimates. In contrast, if it focuses on finding the real best incentive in the exploration phase (by comparing the best incentive to the second best incentive, for example), it is likely to apply the incentives more. This means it would waste the budget on the less effective incentives. From (9), (10), and (11), we have:

$$\beta_{i_1} \sqrt{\frac{\ln\left(1/(1 - l_1^{(t)})\right)}{2u_{i_1}^{(t)}}} + \beta_{i_2} \sqrt{\frac{\ln\left(1/(1 - l_2^{(t)})\right)}{2u_{i_2}^{(t)}}} \leq \delta_{i_2} - \delta_{i_1}. \tag{12}$$

We assume that $(\bar{X}_{i_1}^{(t)} - \delta_{i_1} < \gamma_{i_1})$ and $(\bar{X}_{i_2}^{(t)} - \delta_{i_2} > -\gamma_{i_2})$ are two independent events. This is acceptable because we can prevent a user from participating in more than one group in a period. Thus, the performance of users in different incentives are unrelated to each other. In more detail, in crowdsourcing platforms such as Amazon Mechanical Turk, Clickworker, or Figure Eight, the number of users is large. And, when submitting new tasks we can easily filter out the users who already participated in the project (by using the provided APIs). Even with crowdsourcing projects whose potential number of users is not large or it is difficult to re-recruit users, a small number of users recruited more than once is not likely to change the result significantly. However, a larger number of these might do and hence is not considered in this work. Therefore, the confidence level of both these events occurring is $l_1^{(t)} l_2^{(t)}$. To keep our analysis simple, we choose the same confidence level in (9) and (10), i.e., $l_1^{(t)} = l_2^{(t)} \stackrel{\text{def}}{=} \sqrt{L_h}$ (where $t = 2$). Additionally, despite the fact that the numbers of users on

the worst and best active incentives after period 1 ($u_{i_1}^{(t)}$ and $u_{i_2}^{(t)}$) might be different (because of different group sizes), the target number of sampled users to obtain in this step (i.e., until the end of period 2) is expected to be the same (i.e., $u_{i_1}^{(2)} = u_{i_2}^{(2)} \stackrel{\text{def}}{=} u^{(2)}$). Thus, from (12) we have:

$$u^{(2)} \geq \frac{\ln\left(1/(1 - \sqrt{L_h})\right)\left(\beta_{i_1} + \beta_{i_2}\right)^2}{2\left(\delta_{i_2} - \delta_{i_1}\right)^2} \stackrel{\text{def}}{=} U_2. \tag{13}$$

Since δ_i ($\forall i = 1, \ldots, I$), β_{i_1}, and β_{i_2} are unknown in advance, we use the estimates after the sampling step to approximate these values:

$$\delta_i \approx d_i^{(1)},$$

$$\beta_{i_1} \approx b_{i_1}^{(1)} \stackrel{\text{def}}{=} \max_{1 \leq u \leq u_{i_1}^{(1)}}\left\{X_{i_1,u}^{(1)}\right\} - \min_{1 \leq u \leq u_{i_1}^{(1)}}\left\{X_{i_1,u}^{(1)}\right\}, \text{ and}$$

$$\beta_{i_2} \approx b_{i_2}^{(1)} \stackrel{\text{def}}{=} \max_{1 \leq u \leq u_{i_2}^{(1)}}\left\{X_{i_2,u}^{(1)}\right\} - \min_{1 \leq u \leq u_{i_2}^{(1)}}\left\{X_{i_2,u}^{(1)}\right\}. \tag{14}$$

Therefore, from (13) we have:

$$U_2 \approx \frac{\ln\left(1/(1 - \sqrt{L_h})\right)\left(b_{i_1}^{(1)} + b_{i_2}^{(1)}\right)^2}{2\left(d_{i_2}^{(1)} - d_{i_1}^{(1)}\right)^2}. \tag{15}$$

Similar to the sampling step, this step is also constrained by the budget bound $\epsilon_1 B$. Hence, HAIS uses the approach applied in the sampling step to deal with this (Line 10) by approximating the maximum number of users based on the total cost of applying the active incentives ($\sum_{i \in \mathbf{A}} c_i / g_i$). Based on the new target number of sampled users u_2, each active incentive will be applied followed by an update to its estimate (Lines 13–15).

4.6 The stepped exploitation step

An important benefit of HAIS is that it can consider stopping sooner, i.e., using fewer periods (e.g., 7 days) than the time budget (e.g., 10 days). Actually, the algorithm will stop stepped exploiting when it reaches a certain level of confidence which is referred to as L_s (s is short for stepped exploitation). L_s can be set in advance as a predefined

Fig. 4 Illustration for (11)
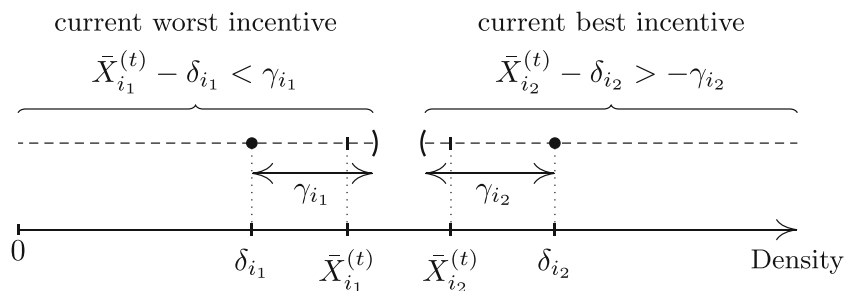
parameter, as it is independent of the actual estimates of the incentives when the algorithm is running. L_s should be set close to 1 (e.g., 90% or 95%) so that we have a high confidence that in the last period (period T) the current best incentive is the best one. The confidence level in finding the real best incentive at the end of period t can be calculated from (12). To keep the algorithm simple, we choose the same confidence level $l_1^{(t)} = l_2^{(t)} = \sqrt{l_t}$ (where $t > 2$). Moreover, we also approximate δ_i ($\forall i = 1, \ldots, I$), β_{i_1}, and β_{i_2} with the estimates so far:

$$\delta_i \approx d_i^{(t)}, \ \beta_{i_1} \approx b_{i_1}^{(t)}, \ \text{and} \ \beta_{i_2} \approx b_{i_2}^{(t)}. \tag{16}$$

Thus, from (12), we have the maximum confidence level in finding the real best incentive at the end of period t:

$$l^{(t)} \approx \left(1 - \exp\left(-\frac{2(d_{i_2}^{(t)} - d_{i_1}^{(t)})^2}{\left(b_{i_1}^{(t)} / \sqrt{u_{i_1}^{(t)}} + b_{i_2}^{(t)} / \sqrt{u_{i_2}^{(t)}} \right)^2} \right) \right)^2. \tag{17}$$

Equation (17) is used before each period in the stepped exploitation step (Lines 17 and 23). to decide whether to continue stepped exploiting or not (the third condition in Line 19).

Additional information can also be used together with the condition about L_s to consider stopping stepped exploiting sooner. Specifically, we can use the number of consecutive periods that the current best incentive has been applied. We refer to this as N_s (a predefined parameter). If in this step, an incentive has been applied consecutively in the last N_s (e.g., 10) periods, this incentive is highly likely to be the real best one. Thus, we can immediately move to the last step (pure exploitation), even if the confidence is still less than L_s because the number of sampled users is not large enough. Therefore, HAIS also uses this information (the fourth condition in Line 19) to decide when to stop stepped exploiting. In this condition, $ns_i^{(t)}$ is the number of consecutive periods that incentive i has been applied at the end of period t.

4.7 The pure exploitation step

In the last period, HAIS exploits the incentives (with the residual budget) by using the *density ordered greedy* approach described in [30], as it is simple and efficient. It is referred to as *pure exploiting* in this work. In detail, it applies the best incentive as many times as it can without exceeding the residual budget. With the remaining budget, it applies the next best incentive, whose cost is not larger than the budget, in the same manner. Note that the incentives to be applied in this step can be the ones which were eliminated after the sampling step (when the residual budget is not

enough to apply any other active incentive). This continues until the budget is not enough to apply any other incentives.

5 A regret bound for HAIS

In this section, we provide a regret bound for the HAIS algorithm. In the development of HAIS, there are several estimates and heuristics (such as (11), (14) and (16)). Hence, in order to analyse the regret bound of the algorithm we assume all these estimates are correct. Also, without loss of generality, we assume that incentive 1 is the best incentive, i.e., the incentive with the highest density. We consider a normalised version of the ISP where the cost of pulling each incentive is the same, and the mean utility of each incentive will be changed accordingly. We adjust the mean values of all incentives so that the incentives have the same cost $c = \frac{1}{I} \sum_{i=1}^{I} c_i$ but the density of each incentive is still unchanged. Specifically, in the normalised ISP each incentive i (with mean μ_i and cost c_i) will have the *adjusted mean* $\widetilde{\mu}_i = \mu_i c / c_i$ and the *normalised cost* c. Thus, the best incentive is now the one with the highest adjusted mean, which is $\widetilde{\mu}_1$. We find a regret bound by measuring the performance of HAIS against the best incentive:

$$Regret_T(\widetilde{\mu}_1) = \sum_{t=1}^{T} \sum_{i=1}^{I} n_i^{(t)} \widetilde{\mu}_1 - \sum_{t=1}^{T} \sum_{i=1}^{I} n_i^{(t)} \widetilde{\mu}_i \tag{18}$$

We have the following theorem:

Theorem 1 *Let the agent follow the HAIS algorithm. Then, the regret of the agent can be bounded by*

$$Regret_T(\widetilde{\mu}_1) \leq \left(\sum_{i=1}^{I'} \frac{\log(\epsilon)\beta_i^2}{-2\gamma^* g_i} \right) \widetilde{\mu}_1 - \sum_{i=1}^{I'} \frac{\log(\epsilon)\beta_i^2}{-2\gamma^* g_i} \widetilde{\mu}_i$$
$$+ \left(\sum_{i=I'+1}^{I} \left[\frac{u_1}{g_i} \right] \right) \widetilde{\mu}_1 - \sum_{i=I'+1}^{I} \left[\frac{u_1}{g_i} \right] \widetilde{\mu}_i + \frac{b_{s_{1,2}}}{c_j} \gamma^*, \tag{19}$$

where $u_1 = \min\left\{ U_1, \frac{\epsilon_1 B}{\sum_{i=1}^{I} c_i / g_i} \right\}$, I' is the number of active incentives after the sampling step, γ^* is the upper bound of the mean's estimate difference in the Hoeffding step with $(1 - \epsilon)$ certainty, $b_{s_{1,2}}$ is the residual budget after steps 1 and 2. Furthermore, if γ^* is chosen as

$$\gamma^* = \sqrt{\frac{\frac{\log(\epsilon)}{-2} \sum_{i=1}^{I'} \frac{\beta_i^2}{g_i} (\widetilde{\mu}_1 - \widetilde{\mu}_i)c}{B}}, \tag{20}$$

then the regret of the agent will be

$$Regret_T(\widetilde{\mu}_1) = \mathcal{O}(\sqrt{B/c}). \tag{21}$$

Proof The full proof is given in Appendix A. $\qquad \square$

Remark 1 From the regret bound in Theorem 1, the regret of HAIS will depend on the parameter γ^*. If γ^* is chosen optimally as shown in Theorem 1, then HAIS is a no-regret algorithm with the regret bound depending linearly on the square root of the number of times each incentive is applied, which is $\sqrt{B/c}$. Intuitively, in the cases when the financial budget B is large, HAIS allows the agent to explore sufficiently every incentive in steps 1 and 2, thus the agent can exploit the best incentive in steps 3 and 4 with high probability. In the cases where T is large, by using $\epsilon-$greedy in the stepped exploitation step, our algorithm can guarantee to find the best incentive by the property of $\epsilon-$greedy [46]. However, in the cases of the ISP, T tends to be small. Thus, applying normal bandit algorithms cannot provide an efficient regret bound. Instead, HAIS can maintain the state-of-the-art regret bound while adapting efficiently to the cases of the ISP where the time budget T is small.

6 Experimental evaluation

To systematically evaluate the performance of HAIS, we use simulations in a wide range of controlled settings. Our aim in so doing is to ascertain the key determinants of performance and how they relate to one another. This is a necessary pre-cursor to real-world deployment. This initial evaluation cannot be undertaken in a real crowdsourcing project as we would have to deploy the project multiple times with different financial budgets, time budgets, number of incentives, and group sizes. Even then we could not guarantee that we have explored the main cases in a comprehensive fashion. In the following, we present the benchmarks (Section 6.1), the experimental settings (Section 6.2), and then discuss the corresponding results (Section 6.3).

6.1 Benchmarks

As the state-of-the-art algorithms discussed in Section 2 are not specifically designed to deal with the time constraints of the ISP, we make a number of modifications to these algorithms.

(1) ϵ-first: This algorithm [30] spends $\epsilon_1 B$ (where ϵ_1 is specified in advance) in the first period to explore by applying the incentives evenly until this budget is exceeded [30]. Then, it spends the subsequent period purely exploiting the best incentives with the residual budget, i.e., $(1 - \epsilon_1)B$ as mentioned in Section 4.7. The purpose of running this algorithm in addition to Stepped ϵ-first (as described below) is to see how effective the stepped exploitation step is.

(2) Stepped ϵ-first (or sϵ-first for short): This algorithm is a modified version of ϵ-first that is designed to run more effectively under a time limit. ϵ-first does not make use of the time budget to exploit effectively, as after the exploration phase, the best incentive might not be the real best one, and this may only be discovered by further exploration. Thus, we apply the stepped exploitation of HAIS to this algorithm to make use of the periods before the deadline to conduct a more effective exploitation (i.e., exploitation together with further exploration). Like HAIS, it spends the last period purely exploiting. An illustration of how this algorithm works is presented in Appendix B.

(3) Stepped fKUBE (or sfKUBE for short): This algorithm [31] applies all the incentives once to obtain initial estimates of the incentives. This can be considered as an initial exploration step. Then, it applies stepped and pure exploitation techniques as per HAIS. The only difference is that Stepped fKUBE uses the upper confidence bounds (UCBs) of the estimates instead of the estimates that HAIS uses. The UCB of incentive i's estimate is:

$$d_i^{*(t)} = \frac{\hat{\mu}_i^{(t)}}{c_i} + \frac{r_{\min} + (r_{\max} - r_{\min})\sqrt{(2\ln u^{(t)})/u_i^{(t)}}}{c_i}. \tag{22}$$

In each period before the last period, it applies the incentive with the highest UCB once followed by an update to the estimate of this incentive.(ucb) In (22), $u^{(t)} = \sum_{i=1}^{I} u_i^{(t)}$ is the total number of users in all incentives until the end of period t and $r_{\min}$ ($r_{\max}$) is the minimum (maximum) density of the incentives, which is specified in Table 1. We will discuss this table in Section 6.2.1. In this step (stepped exploitation), by using the UCBs of the estimates, fKUBE integrates further exploration into the exploitation phase. In fact, as the estimates are uncertain, instead of looking at the estimate of an incentive based only on the current estimate of its expected utility and the cost ($\hat{\mu}_i^{(t)}/c_i$), it also considers the uncertainty of the estimate $\left(\sqrt{(2\ln u^{(t)})/u_i^{(t)}}\right)$. More specifically, when an incentive is applied, this square root term (representing the uncertainty of this incentive's estimate) will decrease. Therefore, regarding this term, the incentives which are applied less (hence, are more uncertain) have more opportunity to be applied in the next period.[12] Finally, in the last period, Stepped fKUBE purely exploits.

---

[12] See [53] for more discussion on UCBs.

(4) Survival of the Above Average (SOAAv): This algorithm [18] applies different incentives from round to round. In each round, it applies the incentives that have estimates above $(1 + \xi)$ times the average of incentives' estimates in the previous period once. The predefined parameter ξ is to help adjust the threshold to eliminate incentives after each period. That means, it only applies the incentives whose estimates are greater than this threshold. If $\xi = 0$, the threshold is the average of the estimates of the incentives. Note that, an eliminated incentive in period i can become active again in period j $(j > i)$ if at the end of period $j - 1$, its estimate is above the threshold. It then updates these incentives' estimates. This happens until the financial budget is exceeded. In the last period, it conducts pure exploitation as in HAIS to exhaust the residual budget.

(5) Exp3: This algorithm [44] maintains a weighted list where each item corresponds to an incentive. The weights are used to randomly choose an incentive in the next periods. After applying an incentive and receiving a utility, the algorithm updates the weight of this incentive based on the received utility. More specifically, at the beginning the weights of the incentives $(w_i^{(1)})$ are all 1. In the first period, the algorithm applies each incentive once to obtain initial estimates of the incentives. Then, it updates the weights of all the incentives. The way Exp3 updates the weights at the end of period 1 is the same as in the other periods before the deadline, which is shown in (24). In period $t = 2, \ldots, T - 1$, the probability of choosing incentive i $(i = 1, \ldots, I)$ is:

$$p_i^{(t)} = (1 - \gamma) \frac{w_i^{(t)}}{\sum_{j=1}^{I} w_j^{(t)}} + \frac{\gamma}{I}, \qquad (23)$$

where $\gamma \in (0, 1]$ is a predefined parameter to specify the level of exploration to be used. Specifically, when $\gamma = 1$, the first term on the right hand side of (23) is 0. Hence, the algorithm ignores the incentives' weights (i.e., it completely explores). When γ is closer to 0,

this term is greater. That means, the probability of choosing an incentive is based more on its weight (i.e., more exploitation). At the end of period t, the received utility $(r_i^{(t)})$ will be used to update the weights of the incentives to prepare for the next period:

$$w_j^{(t+1)} = \begin{cases} w_j^{(t)} \exp\left(\frac{\gamma}{I c_j p_j^{(t)}} \cdot \frac{r_j^{(t)} - r_{\min}}{r_{\max} - r_{\min}} \right), & \text{if } j = i \\ w_j^{(t)}, & \text{otherwise} \end{cases}, \qquad (24)$$

where $r_{\min}$ $(r_{\max})$ is the minimum (maximum) density of the incentives, which is specified in Table 1. In the last period, Exp3 conducts pure exploration as in HAIS.

(6) Optimal: It simply applies the real best incentive all the time. To do so, we have to know the utility means μ_i $(\forall i = 1, \ldots, I)$ in advance, which are unknowable in our practice. Thus, it is unachievable for real-world development.

6.2 Simulation settings

To evaluate the performance of the algorithms we run simulations in seven different settings where the independent variables are financial budget, time budget, number of incentives, standard deviation of the incentives' utilities, and maximum group size. Regarding the latter, we run three settings and in each setting, we draw the group size of each incentive in each simulation from a discrete uniform distribution from 1 to the maximum group size. We will describe these three settings later in the section.

The simulations in these seven settings help us compare the algorithms in terms of performance (i.e., the average density). Based on these simulations, we cannot readily see why one algorithm performs better (or worse) than the others. Therefore, we run other simulations on a representative case so that we can better understand the behaviour of each algorithm (other cases give broadly the same outcomes). Specifically, based on the simulations, we

Table 1 Ranges for Randomisation of the Parameters in the Simulations. All the values are integers and uniformly distributed

Parameter	Symbol	Min value	Max value	Unit
Number of incentives	I	2	20	Incentives
Group sizes	g_i	1	50	Users
Real densities	δ_i	60	90	Utility[13] per £
Utility means	μ_i	60	90	–
Utility stand deviations	σ_i	$0.2\mu_i$	$0.6\mu_i$	–
Financial budget	B	10	100	Times of the round cost
Time budget	T	2	30	Periods

want to examine how the algorithms spread the budget across the phases and steps and over the incentives.

Regarding the seven settings, in the simulations of each setting, the related quantities, i.e., $B, T, I, g_i, c_i, \mu_i, \delta_i \ \forall i = 1, \ldots, I$ (except the corresponding independent variable) are generated uniformly in specific ranges. The ranges of the quantities are shown in Table 1 and will be discussed in more detail in Section 6.2.1.

In terms of the maximum group size settings, we run one setting to examine the performance of the algorithms with different values for the maximum group size. Specifically, in the simulations of this setting, group sizes of the incentives are generated uniformly from 1 to the value of the independent variable. In addition, we also run two more settings in two special cases. Concretely, since the algorithms (excluding `HAIS`) apply the incentives without considering the group sizes, when the group size of the real best (worst) incentive is largest, these algorithms have an advantage (disadvantage) over `HAIS`. For example, if the group size of the real best incentive is largest, by applying the incentives evenly in the exploration phase, ϵ-`first` and `Stepped` ϵ-`first` also partially exploit the best incentive as it has more sampled users on this incentive. However, `HAIS` does not have that exploitation while exploring as in its exploration phase it tries to apply the incentives so that the number of sampled users on each incentive is almost the same. Additionally, by having more sampled users in the exploration phase, ϵ-`first` and `Stepped` ϵ-`first` have a better estimate of the real best incentive and hence they are likely to recognise that this is indeed the real best incentive after exploring. Therefore, we want to investigate how `HAIS` performs compared to other algorithms in these two special cases. In the simulations of these two settings, we keep the group size of the real best (worst) incentive fixed with the value of the independent variable (x). The group sizes of the other incentives are generated randomly from 1 to $x - 1$ (to ensure they are always smaller).

For each value of the independent variable, we run 20,000 simulations to achieve statistically significant results at the 99% confidence level. In Figs. 5–13 and 16, the confidence intervals are small. So, for better image clarity, the error bars representing the confidence intervals are omitted. To better understand the algorithms' behaviours, we run with six incentives where the densities of incentives 1 to 6 are 90, 80, 75, 75, 70, and 60 respectively. That means incentive 1 is the best, while incentive 6 is the worst. In the simulation, the budgets are £3,000 and 10 periods, and the standard deviation of incentive i is $0.4\mu_i \ \forall i = 1 \ldots 6$ (the mean value of the range presented in Table 1 which will be discussed in the next subsection). The group size of each incentive in each period is generated uniformly in the range from 1 to 10. We also run the simulation 20,000 times as with the above-mentioned simulations.

Next, in Section 6.2.1, we detail the ranges of the quantities used for randomisation in the simulations. Then, in Section 6.2.2, we detail the values of the algorithms' predefined parameters. Finally, in Section 6.2.3, we present how the performance of a group is generated in the simulations (based on the performance of the individuals of the group).

6.2.1 Ranges of the quantities for randomisation

The ranges of the quantities are described in Table 1. The values are chosen to represent realistic settings from a number of real crowdsourcing projects. The projects will be presented in the corresponding parameters. As the crowdsourcing projects found in the literature are not run using MABs, based on the figures in these projects (such as budgets or group sizes), we infer the ranges for the related quantities in our simulations. The papers used for inferring the ranges will be stated when possible. In more detail, regarding the *number of incentives*, as will be shown later in Section 6.3, the more incentives the worse the performance of the algorithms becomes. This is reasonable because the more incentives the more budget spent on exploring their effectiveness. Hence, in a real crowdsourcing project, the chosen number of incentives should be as small as possible. For this reason, we choose 20 as the maximum value of I. We can have 20 separate incentives or 5 group sizes with 4 payment structures per group size.

Regarding the *group sizes*, according to the figures from [54], the popular group sizes on Taskcn are from 1 to about 100. However, it is more difficult to recruit many users (for a contest), especially with crowdsourcing projects that are not run on other platforms (such as Amazon Mechanical Turk or Clickworker) and hence they have to recruit users by themselves [29]. Additionally, when users get experience with crowdsourcing contests, they tend to participate in the contests with small group sizes so that they can have a better chance to win the competition [54]. Because of this, the chosen maximum value for group sizes is 50 (instead of 100).

Regarding the *densities and utility means*, since each crowdsourcing project can use a different way to measure the utility (as discussed in Section 3), the range of densities can be very different. In our simulations, we combine both the quantity and quality of the tasks (i.e., number of tasks completed and their corresponding quality) in the metric.

---

[13]In the simulation settings, we know the real density of the worst incentive (which is 60 utility per £). So, to have a better comparison between the algorithms, the effectiveness of each algorithm is measured by the increase in utility over the worst algorithm. The worst algorithm is the algorithm which simply applies the worst incentive as many times as possible. We refer to this increase in utility as *normalised utility*.

So, we choose [60..90] as the possible utility means and [60..90] as the possible density values of the incentives. The maximum difference between the best and worst incentives is 30 but not larger because in real crowdsourcing projects, by using prior knowledge about the projects (if possible) together with existing studies, we can build good-enough incentives. Although some of the designed incentives may be relatively poor (e.g., their densities are 30 or 40), they do not result in a significant difference in the results. Hence, we skip these cases and concentrate on the more challenging settings where performance differences are relatively small. Moreover, to observe the performance of the algorithms more clearly with different values of the independent variables, the density of the best incentive is always 90.

Regarding the *utility standard deviations*, when these values are too small (e.g., $0.05\mu_i$), the algorithms can easily identify the real densities of the incentives. Similarly, when they are too large (e.g., $0.9\mu_i$), it is very challenging for all the algorithms to estimate the incentives, as they need a much higher budget to obtain better estimates. This is infeasible in real crowdsourcing projects, where the budgets are usually limited. Therefore, as the purpose of the simulations is to compare the performance of the algorithms, we use an average range of the standard deviations, that is from $0.2\mu_i$ to $0.6\mu_i$ ($\forall i$).

Regarding the *financial budget*, to allow us to carry out a meaningful performance comparison, the budget should not be too small. If the algorithms do not have a sufficient budget for exploring, then all of their performances will be low. Also, as the number of incentives and group sizes are generated uniformly, to be sure the budget is not too small, its value should be proportional to these quantities. Therefore, we use *round cost* to control the minimum value of the budgets. Here, round cost (denoted by $round\_cost$) is the cost of applying all incentives where each incentive has U_1 sampled users.

According to our calculation, the budgets used for the first crowdsourcing project in [23] (experiment 1: image ordering) and the crowdsourcing project in [25] (the experiment with the word puzzle) are about about 94 and 58 times the round cost. In these studies, as they use individual-based incentives, the round cost is the cost of one user in all treatments of the corresponding experiment. Moreover, since these two crowdsourcing projects are running behavioural experiments, the real crowdsourcing projects might use larger budgets. Thus, we choose the possible range of the generated financial budgets to be from 10 to 200 times the round cost.

This mechanism is applied to the simulations of all the settings except the three related to the maximum group sizes. Choosing a different mechanism for generating financial budgets in the three settings is because we want to investigate the performance of the algorithms with different values of the maximum group sizes. If this mechanism is also applied to the three settings, the trends can be affected by the financial budgets. Actually, when the maximum value of the group sizes is large, with this mechanism, the financial budget is also large. Thus, the budget for exploitation in HAIS, ϵ-first, and Stepped ϵ-first is large. This might affect the general performance of the algorithms. Therefore, in these three settings, we use the above-mentioned generating mechanism with one change. The round cost is replaced with the median value of the range of the group sizes as described in Table 1, that is 25.5. By doing so, different values of the independent variable x (i.e., the maximum group sizes) do not affect the generated financial budgets. Hence, the performance of the algorithms is influenced by x only.

Regarding the *time budget*, as the result of 1 period is uninteresting (i.e., nothing can be learnt), we choose 2 periods as the minimum value of T. We also choose 30 as the maximum value of T. Depending on the characteristics of specific crowdsourcing projects and how long of a period, the most likely time budgets are believed to be in this range. For example, if a period is 1 week, then several (e.g., 8) weeks is a reasonable deadline. Or, if a period is 1 day, then 30 days for the time budget is feasible.

6.2.2 Values of the predefined parameters of the algorithms

We run the algorithms with different values of the predefined parameters and then choose appropriate values for the parameters. For example, with ϵ_1 of ϵ-first, we first run this algorithm with different values (such as 0.05, 0.1, 0.2, 0.3, and 0.4). Then we choose one value that helps ϵ-first perform well in different settings. A similar process is used for the other predefined parameters such as ϵ_2 of Stepped ϵ-first and L_h of HAIS. We can automate the process of choosing appropriate values for these predefined parameters by using Baysian optimisation [55, 56].

As changing these values slightly does not result in a significant difference (i.e., the trends of the algorithms' performance are broadly the same), in Section 6.3, we only present the results on the simulations with the values of the algorithms' predefined parameters as described in Table 2. Regarding the predefined parameters of HAIS, as most of them are self-explanatory and some of them are already discussed in Section 4.3, we do not explain them here.

6.2.3 The model of group performance

In the simulations, we assume that the performance of a group (i.e., the total utility of all users in the group) is proportional to the group size. This means the more

Table 2 Values of the algorithms' predefined parameters in the simulations

Algorithm	Parameter	Value	Description
HAIS	ϵ_1	0.10	Budget limit for exploration.
	ϵ_2	0.50	Budget for stepped exploitation (calculated based on the residual budget).
	ϵ_{greedy}	0.10	The probability of choosing to explore (i.e., selecting a random incentive) in each period of stepped exploitation.
	U_1	20	Target number of sampled users to obtain after the first (sampling) period.
	L_e	90%	Confidence level to calculate confidence intervals of the incentives' estimates for eliminate ineffective incentives.
	L_h	50%	Confidence level to stop exploring.
	L_s	90%	Confidence level to stop stepped exploiting.
	N_s	5	Maximum number of consecutive periods that an incentive is applied in the stepped exploitation step.
ϵ-first	ϵ_1	0.10	Budget limit for exploration.
Stepped	ϵ_1	0.10	Budget limit for exploration.
ϵ-first	ϵ_2	0.50	Budget for stepped exploitation (calculated based on the residual budget).
Exp3	γ	0.50	Exploration factor
	ϵ_2	0.50	Budget for stepped exploitation (calculated based on the residual budget).
Stepped fKUBE	ϵ_2	0.50	Budget for stepped exploitation (calculated based on the residual budget).
SOAAv	ξ	0	$\xi = 0$ means the incentives to be applied in a period are the ones whose estimates are above the average of the estimates of the ones in the previous period.

users there are in a group, the better the performance of the whole group. In the literature, there are very few papers investigating the performance of a group of users in crowdsourcing contests. This assumption is based on an empirical study conducted by [49]. In their work, they investigate the data collected from 99designs, a crowdsourcing platform where users submit their designs and compete with others for a financial reward. They found that the quality of the designs in a contest is almost linear in the number of users who participated in the contest.

6.3 Results

In general, HAIS performs best in most cases (Figs. 5–11). In more detail, HAIS performs better with a larger financial budget (Fig. 5), with a looser deadline (Fig. 6), with fewer

Fig. 5 Performance of algorithms for different financial budget sizes
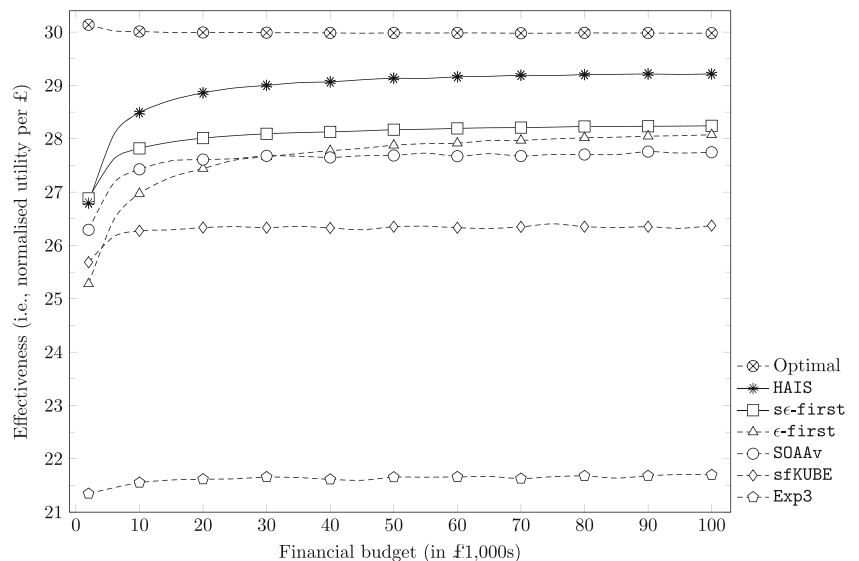
Fig. 6 Performance of algorithms for different time budget values
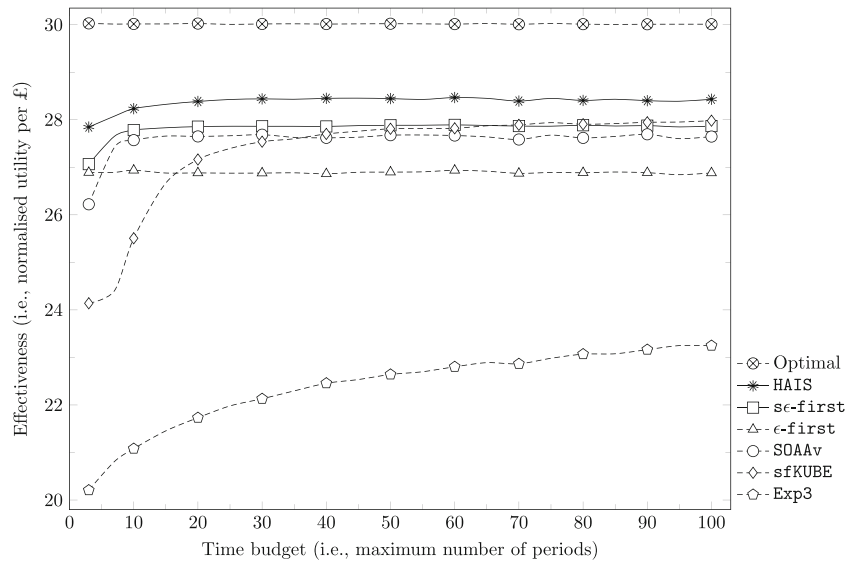
Fig. 7 Performance of algorithms for different numbers of incentives
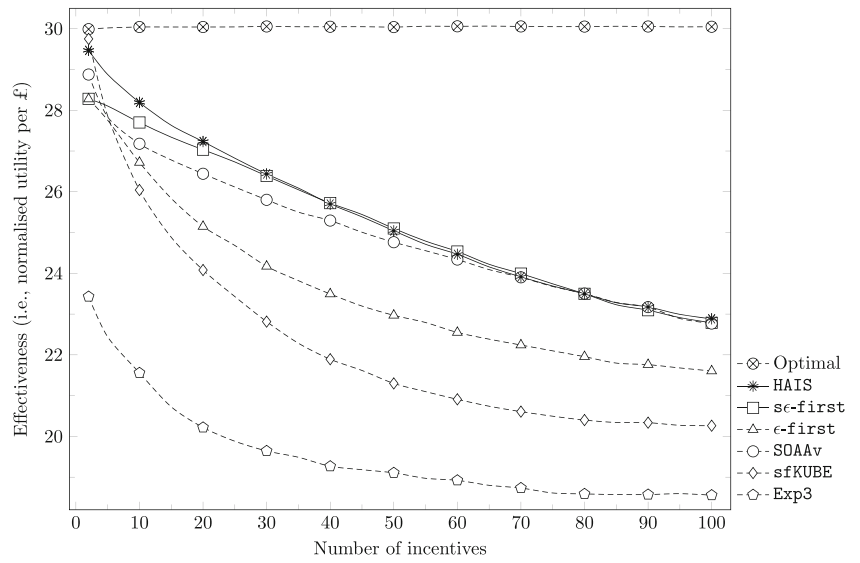
Fig. 8 Performance of algorithms for different values of the standard deviation of the incentives' utilities
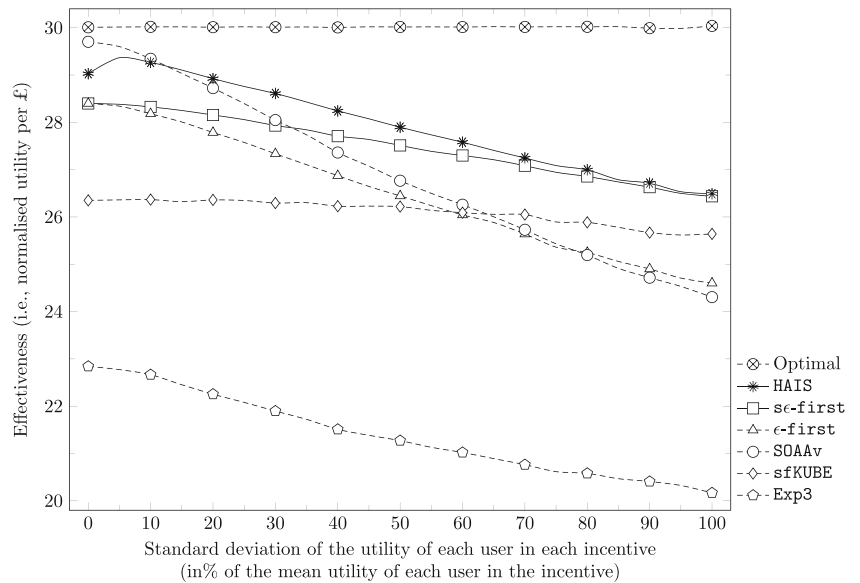
Fig. 9 Performance of
algorithms for different values
of maximum group size
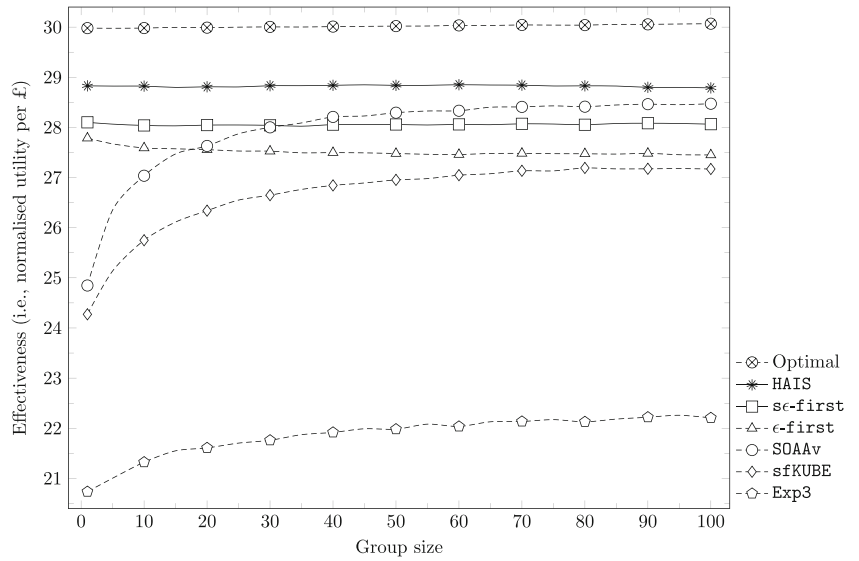
Fig. 10 Performance of
algorithms for different values of
maximum group size on the best
incentive. Group size of the best
incentive is fixed with the value
of the independent variable, x,
while group sizes of the other
incentives are generated
randomly from 1 to $x-1$
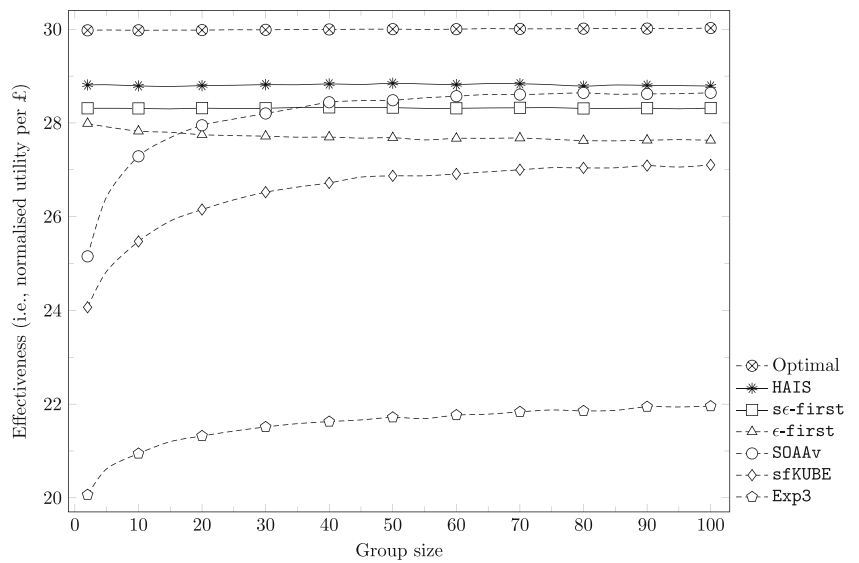
Fig. 11 Performance of
algorithms for different values
of maximum group size on the
worst incentive. Group size of
the worst incentive is fixed with
the value of the independent
variable, x, while group sizes of
the other incentives are
generated randomly from 1 to
$x-1$
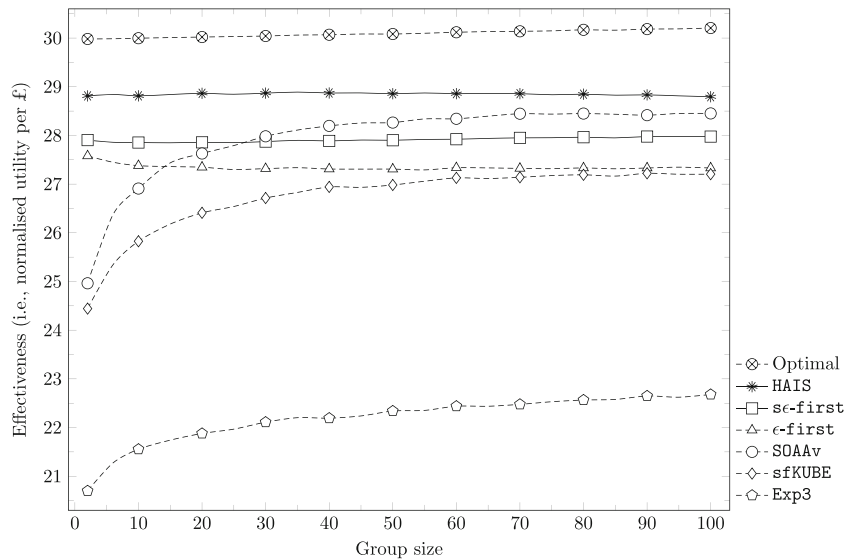
incentives (Fig. 7), and with smaller values of the standard deviation of the incentives' utilities (Fig. 8). Its performance is reasonably stable with different group sizes (Fig. 9), even when the group size of the best incentive is the largest, i.e., when other algorithms (especially `Stepped ε-first`) have an advantage over `HAIS` (Fig. 10).

Additionally, as shown in Fig. 9, when all incentives are individual-based (i.e., their group sizes are all one), `HAIS` performs much better than the benchmarks. This emphasises the performance of `HAIS` in traditional settings where the group-based nature of the arms is omitted.

Moreover, as we can see, `Stepped ε-first` performs much better when the group size of the best incentive is the largest (Fig. 10) than when the group size of the worst incentive is the largest (Fig. 11). The difference is clearer when the maximum group size of the best/worst incentive becomes larger. This is because in Fig. 10 they are likely to have more sampled users in the best incentive. Hence, they can quickly identify this incentive. In both settings, `HAIS` remains almost at the same level of performance.

The reason that `HAIS` can do this effectively is that it has (1) a better exploration-exploitation strategy together with (2) an efficient way of using the time budget in the exploitation phase, and (3) an effective approach for spending more of the budget on highly effective incentives in the exploration phase. We will discuss each of these issues in the following subsections. Then, we will continue with effective ways to use `HAIS` in a specific crowdsourcing project.

However, as in general `Exp3` does not perform well and does not relate to the analysis, we first discuss its performance here and will not consider this algorithm in the remaining subsections. Specifically, `Exp3` does not perform well in any settings (Figs. 5–11). This is because choosing the incentives randomly based on their weights does not work well when the time budget is small. As reflected in Fig. 6, the performance of `Exp3` becomes better when the time budget becomes larger. Yet, in most crowdsourcing projects, the time budgets are usually not large (e.g., several days or months instead of several years). Additionally, with respect to `ε-first`, as the purpose of running this algorithm is to examine the effectiveness of the stepped exploitation step, we only discuss this algorithm in Section 6.3.2 when explaining the importance and the usage of stepped exploitation.

6.3.1 Exploration-exploitation balance

Regarding the exploration-exploitation strategy, as both financial and time budgets are limited in the ISP, an algorithm that takes advantage of the budgets can enhance the overall performance significantly. That is, sufficient exploration should be conducted to identify highly effective

incentives so that the algorithm has enough budget and time to exploit these incentives effectively.

In general, `Stepped fKUBE`'s performance is low. This is because one round for initial exploration is not enough to have good estimates for the next step (stepped exploitation). Actually, as can be seen from Figs. 6 and 9, the performance of this algorithm improves significantly when the time budget or the group sizes become large. This is due to the more time available for the algorithm to identify the best incentive. Also, with larger group sizes, it has more sampled users, and thus the initial estimates become better.

In most cases, `Stepped ε-first` performs better than `Stepped fKUBE` (Figs. 5–11). This is because `Stepped ε-first` spends more of its budget (to have more rounds) for exploring (which is identified by ϵ_1); so it has better estimates of the incentives. However, the performance of `Stepped ε-first` depends on choosing an appropriate value of ϵ_1.

On the other hand, as `HAIS` uses Hoeffding's inequality, it is more flexible in determining an appropriate budget for exploration. Actually, Fig. 12 shows that when the budget for the crowdsourcing project (B) is large, instead of using all $\epsilon_1 B$ as in `Stepped ε-first`, `HAIS` tends to use less of the budget (than `Stepped ε-first`) to explore. Note that although less of the budget is used for exploring, the total cost for applying the best incentive in the exploration phase tends to be larger than that of `Stepped ε-first`. This will be discussed in detail in Section 6.3.3.

6.3.2 Taking advantage of the time budget

By comparing the performance of `Stepped ε-first` with the original `ε-first`, we can see that stepped exploitation helps take advantage of the time budget, and
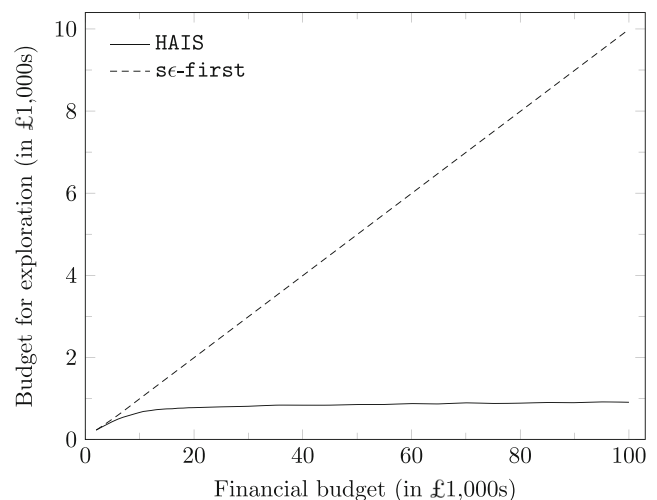
Fig. 12 Average Budget Used for Exploration. This is the corresponding result of the simulations shown in Fig. 5
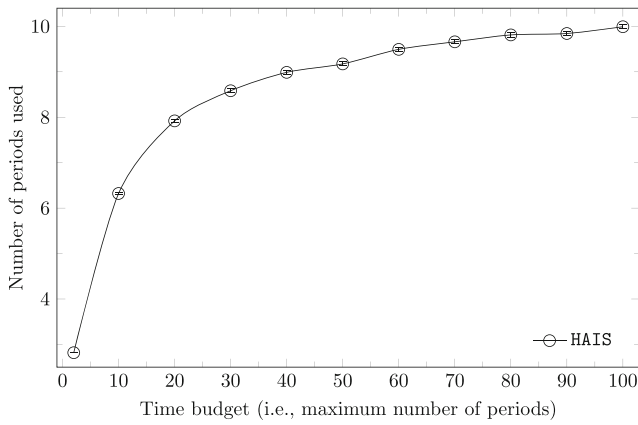
Fig. 13 Number of Periods Used by HAIS. This is the corresponding result of the simulations shown in Fig. 6

hence improves the overall performance of the algorithm significantly. As Figs. 5–11 show, Stepped ϵ-first performs significantly better than ϵ-first, especially when the budget is not large or when the time budget is large. Specifically, in Fig. 5, with low budgets (e.g., from £1,000 to £10,000), ϵ-first does not explore sufficiently;

so, its performance is rather low. Meanwhile, although with the same budgets (i.e., not exploring enough in the exploration phase), Stepped ϵ-first performs much better, as it makes use of the time budget to conduct further exploration while exploiting the incentives. In Fig. 6, this difference in performance between the two algorithms is clearer when the time budget is large (e.g., more than 10 periods). As shown in this figure, since ϵ-first always uses two periods, its performance is almost the same with different values of the time budget.

Although using the same exploitation mechanism, HAIS makes use of stepped exploitation better than Stepped ϵ-first (Figs. 5–11). This is especially the case when the financial budget is large (Fig. 5). In particular, as Stepped ϵ-first has more exploration rounds when the financial budget becomes larger, after the exploration phase, it can identify the highly effective incentives better (i.e., the estimated best incentive is likely to be the real best incentive). Thus, the effect of stepped exploitation on Stepped ϵ-first becomes smaller. Note that, by doing this, Stepped ϵ-first also wastes the budget on applying ineffective incentives in the exploration phase. This is shown in Fig. 5 where ϵ-first's effectiveness

Fig. 14 Cost distribution over the incentives across the phases of each algorithm
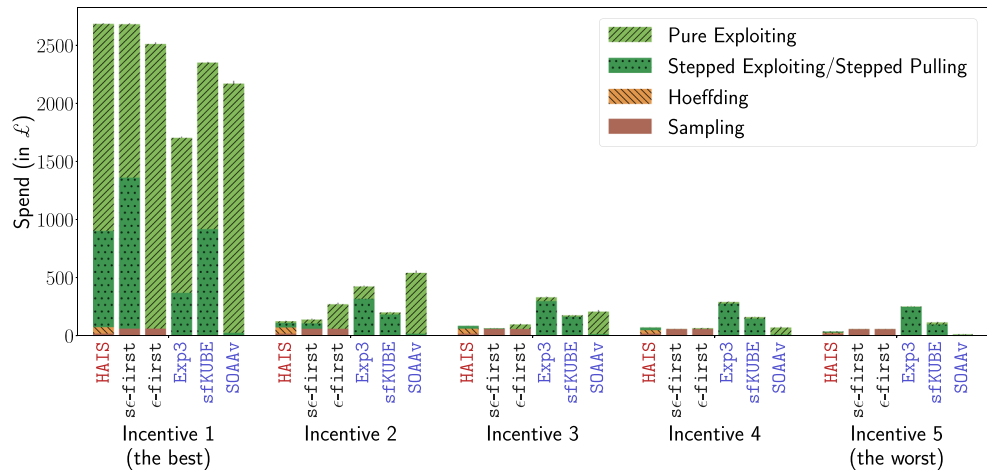
(a) The whole figure
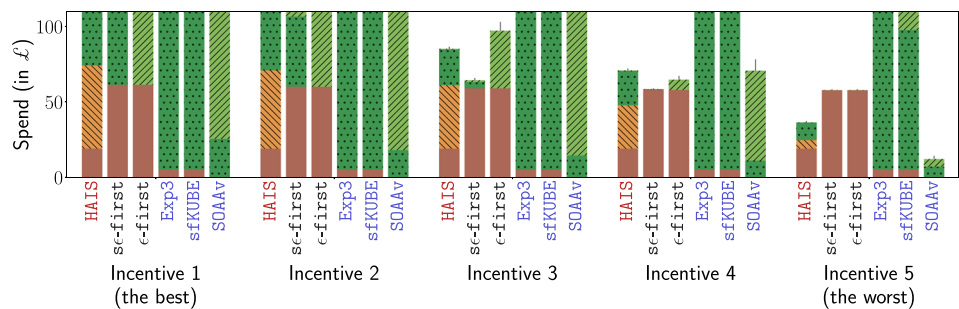
(b) A zoom of (a)

approaches that of `Stepped ε-first` when the budget size becomes larger.

In addition, as discussed in Section 4.6, `HAIS` is able to stop stepped exploiting sooner without significantly affecting the results. Hence, setting a loose deadline is better for its performance as it has enough time to conduct stepped exploitation effectively. To this end, Fig. 13 shows the average number of periods used by `HAIS` in the setting corresponding to Fig. 6. This figure shows that although the time budget is large, `HAIS` tends to use a lot less of it. This suggests that when applying the algorithm to a real crowdsourcing project, if the time is not very important, it is better to set a longer deadline. The algorithm will then automatically select an appropriate time to stop.

6.3.3 Effective elimination

By eliminating clearly ineffective incentives right after having initial estimates and before conducting more exploration, `HAIS` can distinguish highly effective incentives more quickly. The advantage of elimination is that it has more of the budget to continue exploring these incentives (to find the real best one) in the Hoeffding step. Because of this, the Hoeffding step can be considered as not only exploring but also partially exploiting, as it applies only highly effective incentives. The effectiveness of the elimination is shown in Figs. 14 and 15. In more detail, Fig. 14a shows that, compared with other algorithms, `HAIS` spends more of its budget on the best incentive (incentive 1) and less of

Fig. 15 Cost distribution across the periods incurred by each algorithm
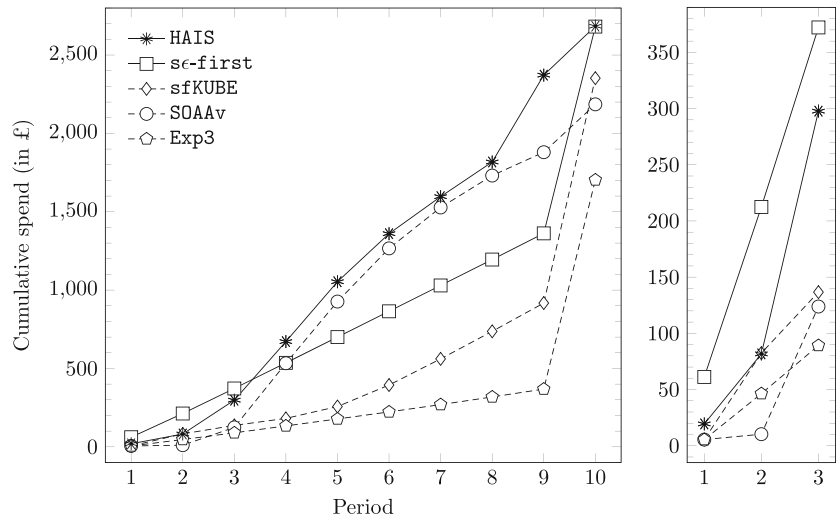
(a) Total spent on the real best incentive. The right graph is a zoom of the left one on the first three periods.
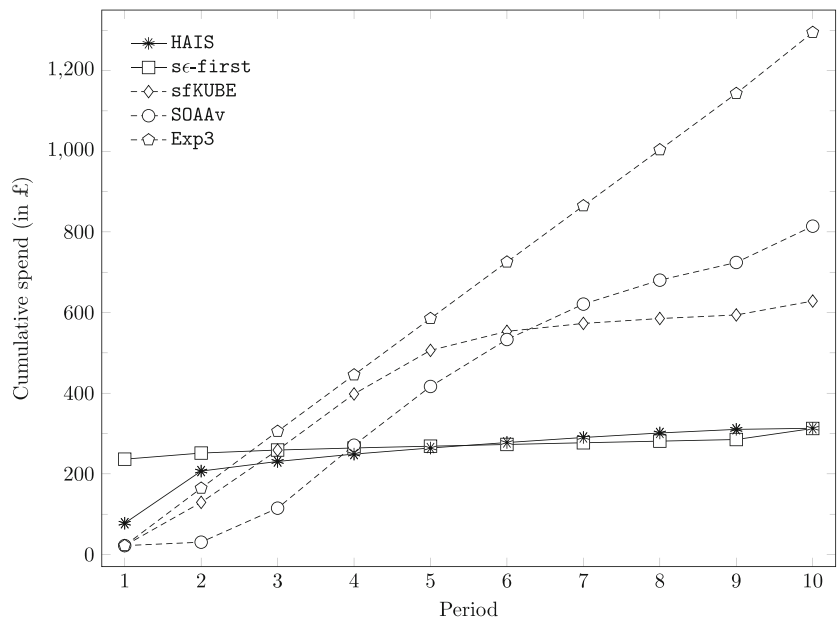
(b) Total spent on the other incentives

its budget on the others. By looking more closely at how the cost is distributed over the incentives across the phases of HAIS (Fig. 14b), we can see that after the sampling step, HAIS identifies ineffective incentives effectively. This figure therefore clearly shows that in the Hoeffding step, HAIS spends more of its budget on highly effective incentives. In contrast to this, Stepped ϵ-first spends the same amount of budget to explore each incentive. This helps HAIS not only partially exploit highly effective incentives while exploring, but also increasing the chance of identifying the real best incentive in the exploitation phase (as the best incentives are likely to be applied more than the others). Indeed, by looking at how the cost is distributed across the periods, we can see that HAIS spends the most on the real best incentive in all exploitation periods, i.e., the periods in the exploitation phase, including the last period (Fig. 15a). Additionally, Fig. 15b shows that HAIS spends less than Stepped ϵ-first on ineffective incentives in all periods. Note that Stepped ϵ-first uses only one period to explore, while HAIS uses two periods. So, Stepped ϵ-first starts exploiting one period sooner than HAIS. Therefore, when comparing the total spent until the end of a certain period (i) of HAIS in the exploitation phase, we need to compare it with that until the end of period $i-1$ of Stepped ϵ-first. For example, we need to compare the total spent from period 1 to period 3 of HAIS with that of periods 1 and 2 of Stepped ϵ-first.

Although using an elimination technique like HAIS, SOAAv does not perform well. Specifically, Fig. 15 shows that SOAAv under-explores the incentives, especially the highly effective ones in the first (e.g., 3) periods. This results in exploiting ineffective incentives in the remaining periods. More specifically, in the first periods SOAAv eliminates the incentives based on the estimates so far (of the incentives' densities). However, in these periods, the algorithm does not have enough sampled users to make good elimination

decisions. Hence, the real best incentive may be eliminated with a probability that is not insignificant. Therefore, in the later periods (e.g., from period 4 to period 9), SOAAv tends to apply the ineffective incentives much more than HAIS and Stepped ϵ-first (Fig. 15). One exception is that SOAAv performs better than HAIS in the case when the difference in the effectiveness of the incentives is small. In detail, Fig. 8 shows that when the standard deviation of the utility of each incentive is less than about 20 per cent of the mean utility of the incentive, SOAAv has slightly higher overall utility than HAIS. The reason is that right after the first period, the estimate of the real best incentive is clearly better than those of the other incentives. Thus, it is likely that the estimated densities of the incentives other than the best one are smaller than the average. Hence, in the remaining periods, these incentives will be eliminated. However, in crowsourcing projects the performance of users tends to be large from user to user depending on their motivations. So, the standard deviations tend to be not too small as in this case. Therefore, we do not include this case in the simulations. Instead, we focus on more realistic cases where the differences are large enough.

6.4 Practical usage of the HAIS algorithm

The above-mentioned results suggest several guidelines for using the HAIS algorithm in practice. First, the larger the budget, the better (Fig. 5). It is reasonable that when the budget is larger, HAIS can spend more on exploring the incentives so that it can identify the best incentives before exploiting.

Second, the fewer incentives, the better (Fig. 7). Specifically, when there are more incentives, HAIS has to spend more of the budget exploring ineffective incentives. But, as the requesters might be uncertain about the effectiveness of the incentives in specific crowdsourcing

Fig. 16 Performance of HAIS for Different Budget Ratios. Budget ratio is $B/round\_cost$. This is the corresponding result of the simulations shown in Fig. 5
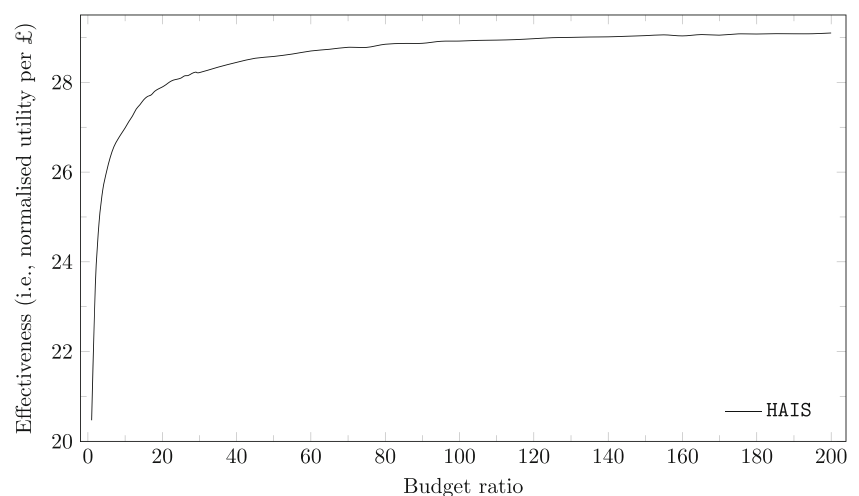
projects, they may not have good reasons to eliminate some chosen candidate incentives so as to improve the overall performance of HAIS. Therefore, Fig. 16 can be used to have a clearer view of how the current number of candidate incentives affects the overall performance. Indeed, the performance of HAIS increases significantly when the budget ratio (i.e., B/round_cost) is from 1 to about 10. After that, it still improves, but slowly. This suggests that the budget should be at least 10 times the round cost. Based on this, with a given financial budget, we can easily determine an appropriate maximum number of incentives.

Third, the time budget should be large enough (e.g., from 15 to 20 periods), but does not need to be very large (e.g., 100 periods) so that HAIS has enough time to conduct stepped exploitation effectively (Fig. 6). Also, if the time budget is set to be larger than necessary, the algorithm will choose an appropriate time to stop.

Fourth, the runtime of the HAIS algorithm to select incentives in a period depends on the number of incentives and the time budget. In the above-mentioned settings, the runtime is less than a second on a desktop computer (2.2 GHz quad-core processor and 16GB internal memory). So, it is feasible for an autonomous agent to quickly identify an appropriate applying policy.

7 Conclusions and future work

We discussed the incentive selection problem and outlined an approach that helps requesters in crowdsourcing projects with a fixed budget maximise their utility. Then, we formalised the problem as a batched 2d-budgeted group-based MAB and introduced an algorithm (HAIS) to solve this effectively. Our algorithm is adaptive and performs efficiently in a wide range of different cases without the need to tune its predefined parameters. Although HAIS is specifically designed for incentives in the form of contests, it can also be used with other types of incentives where the group size is 1 (i.e., there are no contests, such as paying for performance or using bonuses). HAIS significantly outperforms the state-of-the-art approaches in simulations. Additionally, our results also suggest several guidelines for using this algorithm in practice. Regarding other applications of our work, the model proposed and the algorithm developed can be applied in other domains with a group-based nature such as in schools, companies, or organisations (i.e., finding the most effective groups of students or employees to work or study together).

Although HAIS is an important initial step towards solving the incentive selection problem, there are a number of areas of further work. From a practical perspective, we have systematically explored the key determinants of the behaviour in a series of controlled experiments. Such

experiments are a necessary first step to understanding this complex design space. They help us discover the key influences on behaviour and performance. These insights can then be deployed in real-world environments and applications. This is a significant undertaking, but our results provide an excellent foundation for this work. From a more conceptual perspective, there are a number of areas to explore. First, our current model assumes that time steps are homogeneous and a new incentive can be started only when all previous ones have completed. However in some real world settings, the durations of the incentives (e.g., the time to run a contest) might be heterogeneous and variable. The difference in the durations might be large when some incentives are individual-based (e.g., paying for performance) and some others are contests with large group sizes (e.g., 20 users). So, within a given period, groups that finished early will have to wait until all other groups in the period are finished so that the algorithm can move to the next period. Thus, addressing this limitation would shorten waiting times and thereby the total time used by the algorithm. Additionally, this could improve the overall performance as the algorithm has more time to conduct stepped exploitation, especially when the time budget is limited.

A second issue is that the model assumes that the cost of applying an incentive is the same at all times. This may be limiting in more general settings. To motivate top users to continue performing tasks, for example, requesters in crowdsourcing projects might use contests whose prize values depend on the performance of the winners. For instance, instead of paying a fixed £3 to the best user, they could pay from £2 to £4 to the best user depending on the number of tasks completed by this user. This might encourage the best user to do tasks even whey they have a steady top position in the leader board. Moreover, some incentives are inherently designed with variable payment such as paying for performance [23] or using bonuses [25]. Actually, in paying for performance, the more tasks a user completes the more money they earn. And in using bonuses, the bonuses provided depend on the algorithms used and might be different at different times. Therefore, expanding the model to cover the case of variable costs of applying an incentive will provide requesters with more options in choosing incentives to be used in the ISP.

To cope with these limitations, in the ongoing and future work, we will consider using other approaches such as MABs with delayed feedback or reinforcement learning. In more detail, to deal with the homogeneous time steps, MABs with delayed feedback [57] may be a good approach. They focus on MABs where the feedback (i.e., utility) of applying an incentive is not known immediately. By using this approach, in a time step, we do not have to wait for the feedback of all the incentives (pulled

in this step). We can continue applying other incentives and consider the feedback of incentives being applied as delayed. Additionally, reinforcement learning [53] appears to be a promising approach as it can deal with not only the homogeneous and variable time steps but also the variable costs of the incentives. This is because reinforcement learning is designed to work with learning with delayed feedback and in a non-stationary environment (i.e., variable pulling costs).

Appendix A: Proof of Theorem 1

In the sampling step (step 1), since each incentive i will be applied $n_i^{(1)} = [u_1/g_i]$ times (where $u_1 = \min\left\{U_1, \frac{\epsilon_1 B}{\sum_{i=1}^{I} c_i/g_i}\right\}$), the regret in this step is:

$$Regret_{s_1}(\widetilde{\mu}_1) = \left(\sum_{i=1}^{I} n_i^{(1)}\right)\widetilde{\mu}_1 - \sum_{i=1}^{I} n_i^{(1)}\widetilde{\mu}_i. \quad (25)$$

After the sampling step, suppose we only have I' incentives left to move to the Hoeffding step (step 2). That is, $I - I'$ incentives are eliminated with L_e level of confidence. We can calculate the probability of the best incentive (incentive 1) being eliminated after the sampling step. Suppose there exists incentive j such that $d_{1,upper}^{(1)} < d_{j,lower}^{(1)}$. We then have:

$$P\left(\widetilde{\mu}_1 \le d_{1,upper}^{(1)}\right) = 1 - \frac{1 - L_e}{2}$$

$$P\left(d_{j,lower}^{(1)} \le \widetilde{\mu}_j\right) = \frac{1 + L_e}{2}$$

$$\implies P\left(\widetilde{\mu}_1 \le d_{1,upper}^{(1)} \text{ and } d_{j,lower}^{(1)} \le \widetilde{\mu}_j\right) = \left(\frac{1 + L_e}{2}\right)^2$$

$$\implies P\left(d_{j,lower}^{(1)} \le d_{1,upper}^{(1)}\right) \ge \left(\frac{1 + L_e}{2}\right)^2$$

$$\implies P\left(d_{1,upper}^{(1)} < d_{j,lower}^{(1)}\right)$$

$$= 1 - P\left(d_{1,upper}^{(1)} \ge d_{j,lower}^{(1)}\right) \le 1 - \left(\frac{1 + L_e}{2}\right)^2.$$

Since $\widetilde{\mu}_1 \ge \widetilde{\mu}_j$ and $P(B) \ge P(B)P(A|B) = P(A)P(B|A) = P(A)$ where A and B are the events $\left(\widetilde{\mu}_1 \le d_{1,upper}^{(1)} \text{ and } d_{j,lower}^{(1)} \le \widetilde{\mu}_j\right)$ and $\left(d_{j,lower}^{(1)} \le d_{1,upper}^{(1)}\right)$ respectively. Thus, with probability

$1 - \left(\frac{1+L_e}{2}\right)^2$ (i.e., a small probability), the best incentive (incentive 1) is eliminated after step 1 .

Now we consider step 2 when we use the Hoeffding's inequality for the remaining I' incentives, which includes the best incentive with high probability. In HAIS, using a small γ and increasing the number of times the incentive is applied to reduce the probability bound (i.e., $\exp\left(-2u_i(t)\gamma^2/\beta_i^2\right)$) is the same as using a small probability bound and increasing the number of samples to reduce the number γ_t. Suppose that after step 1, with a small probability bound ϵ (i.e., $\epsilon = \exp\left(-2u_i(t)\gamma^2/\beta_i^2\right)$), the difference between the mean and the estimate of each incentive i will have lower bound and upper bound of $(-\gamma_i^1, \gamma_i^1)$. We will apply each incentive more times to make sure that with the same probability, the upper bound of each incentive i will be less than a certain level of accuracy, γ^*. Therefore, after the Hoeffding step, with high probability (i.e., $1 - 2\epsilon$), we have the estimate of the mean value of each incentive i as follows:

$$|\mathring{\mu}_i - \widetilde{\mu}_i| \le \gamma^* \quad \forall i \in [1, I'].$$

Thus, the regret in step 2 will be:

$$Regret_{s_2}(\widetilde{\mu}_1) = \left(\sum_{i=1}^{I'} n_i^{(2)}\right)\widetilde{\mu}_1 - \sum_{i=1}^{I'} n_i^{(2)}\widetilde{\mu}_i, \quad (26)$$

where $n_i^{(2)}$ is the number of times incentive i is applied in step 2, which can be specified by:

$$n_i^{(2)} = \frac{\log(\epsilon)\beta_i^2}{-2\gamma^* g_i} - n_i^{(1)}.$$

After this step, the predicted best incentive j (i.e., $j = \arg\max_{i \in [1, I']} \mathring{\mu}_i$), will have the following property with high probability (i.e., $(1 - \epsilon)^2$):

$$\widetilde{\mu}_1 - \widetilde{\mu}_j = \widetilde{\mu}_1 - \mathring{\mu}_1 + \mathring{\mu}_j - (\widetilde{\mu}_j - \mathring{\mu}_j) \le \gamma^* + 0 + \gamma^* = 2\gamma^*. \quad (27)$$

In the stepped exploitation step (step 3), the algorithm results in reducing the gap $(\widetilde{\mu}_j - \mathring{\mu}_j)$ (i.e., the current predicted best incentive) by increasing the number of samples of incentive j. However, if the best incentive does not appear in the current predicted best incentive set, then step 3 does not improve the estimate of the best incentive. Thus the best bound which our algorithm guarantees in steps 3 and 4 (pure exploitation) will be

$$\widetilde{\mu}_1 - \widetilde{\mu}_j \le \widetilde{\mu}_1 - \mathring{\mu}_1 + \mathring{\mu}_j - (\widetilde{\mu}_j - \mathring{\mu}_j) \le \gamma^* + 0 + 0 = \gamma^*, \quad (28)$$

when we get the exact estimate for the predicted best incentive j (e.g., $\widetilde{\mu}_j - \mathring{\mu}_j = 0$). Let $b_{1,2}$ be the residual budget after steps 1 and 2. Then the regret in steps 3 and 4 will be bounded by:

$$Regret_{s_{3,4}}(\widetilde{\mu}_1) = \frac{b_{1,2}}{c_j}\widetilde{\mu}_1 - \frac{b_{1,2}}{c_j}\widetilde{\mu}_j \le \frac{b_{1,2}}{c_j}\gamma^*. \quad (29)$$

Note that in the above regret, we ignore the fact that the estimate $\mathring{\mu}_j$ will slowly converge to $\widetilde{\mu}_j$ in step 3 as the difference in Inequality (25) will always be $O(\gamma^*)$. From (25) and (26) and Inequality (29) we have:

$$Regret_T(\widetilde{\mu}_1) = Regret_{s_1}(\widetilde{\mu}_1) + Regret_{s_2}(\widetilde{\mu}_1) + Regret_{s_{3,4}}(\widetilde{\mu}_1)$$

$$\leq \left(\sum_{i=1}^{I} n_i^{(1)}\right)\widetilde{\mu}_1 - \sum_{i=1}^{I} n_i^{(1)}\widetilde{\mu}_i +$$

$$\left(\sum_{i=1}^{I'} \frac{\log(\epsilon)\beta_i^2}{-2\gamma^* g_i} - n_i^{(1)}\right)\widetilde{\mu}_1$$

$$-\sum_{i=1}^{I'}\left(\frac{\log(\epsilon)\beta_i^2}{-2\gamma^* g_i} - n_i^{(1)}\right)\widetilde{\mu}_i + \frac{b_{1,2}}{c_j}\gamma^*,$$

or

$$Regret_T(\widetilde{\mu}_1) \leq \left(\sum_{i=I'+1}^{I}\left[\frac{u_1}{g_i}\right]\right)\widetilde{\mu}_1 - \sum_{i=I'+1}^{I}\left[\frac{u_1}{g_i}\right]\widetilde{\mu}_1$$

$$+ \left(\sum_{i=1}^{I'}\frac{\log(\epsilon)\beta_i^2}{-2\gamma^* g_i}\right)(\widetilde{\mu}_1 - \widetilde{\mu}_i) + \frac{b_{1,2}}{c_j}\gamma^*, \quad (30)$$

where $u_1 = \min\left\{U_1, \frac{\epsilon_1 B}{\sum_{i=1}^{I} c_i/g_i}\right\}$.

Inequality (30) shows that the total regret $Regret_T(\widetilde{\mu}_1)$ will depend on the parameter γ^* (i.e., the accuracy of the Hoeffding step). With the normalised problem, the number of times an incentive will be applied is B/c, thus the regret in Inequality (30) can be rewritten as:

$$Regret_T(\widetilde{\mu}_1) \leq \left(\sum_{i=I'+1}^{I}\left[\frac{u_1}{g_i}\right]\right)\widetilde{\mu}_1 - \sum_{i=I'+1}^{I}\left[\frac{u_1}{g_i}\right]\widetilde{\mu}_1$$

$$+ \frac{\log(\epsilon)}{-2}\sum_{i=1}^{I'}\frac{\beta_i^2}{g_i}(\widetilde{\mu}_1 - \widetilde{\mu}_i)\frac{1}{\gamma^*}$$

$$+ \left(\frac{B}{c} - \sum_{i=1}^{I'}\frac{\log(\epsilon)\beta_i^2}{-2\gamma^* g_i}\right)\gamma^*, \quad (31)$$

or

$$Regret_T(\widetilde{\mu}_1) \leq \left(\sum_{i=I'+1}^{I}\left[\frac{u_1}{g_i}\right]\right)\widetilde{\mu}_1 - \sum_{i=I'+1}^{I}\left[\frac{u_1}{g_i}\right]\widetilde{\mu}_1$$

$$+ \frac{\log(\epsilon)}{-2}\sum_{i=1}^{I'}\frac{\beta_i^2}{g_i}(\widetilde{\mu}_1 - \widetilde{\mu}_i)\frac{1}{\gamma^*}$$

$$+ \left(\frac{B}{c}\right)\gamma^* - \sum_{i=1}^{I'}\frac{\log(\epsilon)\beta_i^2}{-2g_i}. \quad (32)$$

When B is large, the following part in Inequality (32) will be negligible:

$$\left(\sum_{i=I'+1}^{I}\left[\frac{u_1}{g_i}\right]\right)\widetilde{\mu}_1 - \sum_{i=I'+1}^{I}\left[\frac{u_1}{g_i}\right]\widetilde{\mu}_1 - \sum_{i=1}^{I'}\frac{\log(\epsilon)\beta_i^2}{-2g_i}. \quad (33)$$

This is because this part does not depend on B. Thus, Inequality (30) can be rewritten as:

$$Regret_T(\widetilde{\mu}_1) = \mathcal{O}\left(\frac{\log(\epsilon)}{-2}\sum_{i=1}^{I'}\frac{\beta_i^2}{g_i}(\widetilde{\mu}_1 - \widetilde{\mu}_i)\frac{1}{\gamma^*} + \left(\frac{B}{c}\right)\gamma^*\right). \quad (34)$$

Denote $A = \frac{\log(\epsilon)}{-2}\sum_{i=1}^{I'}\frac{\beta_i^2}{g_i}(\widetilde{\mu}_1 - \widetilde{\mu}_i)$, then by setting γ^* equal to

$$\gamma^* = \sqrt{\frac{\frac{\log(\epsilon)}{-2}\sum_{i=1}^{I'}\frac{\beta_i^2}{g_i}(\widetilde{\mu}_1 - \widetilde{\mu}_i)c}{B}} = \sqrt{\frac{Ac}{B}},$$

the regret of HAIS will be:

$$Regret_T(\widetilde{\mu}_1) = \mathcal{O}\left(\sqrt{\frac{AB}{c}}\right). \quad (35)$$

Equation (35) shows that the total regret $Regret_T(\widetilde{\mu}_1)$ is sublinear in $\frac{B}{c}$, which is the average number of times the incentives are applied.

Appendix B: An Illustration of how `Stepped ε-first` Works

Figure 17 shows an example of how Stepped ε-first runs in a simple case. The setting in this figure is the same as the one in Fig. 2. In the exploration phase (day 1) of this example, as it does not look at the group sizes (as per HAIS), it applies the incentives evenly (4 times each). After this period, suppose that the estimate of incentive 1 is higher than those of the other incentives. So, based on these estimates, Stepped ε-first identifies that incentive 1 is the best one. Compared with ε-first, it is better as instead of applying incentive 1 (the best incentive) 12 times with the residual budget of $48 as in ε-first, it distributes half ($\epsilon_2 = 0.50$) of this budget (that is $24) equally across the next three periods ($8 each on days 2, 3, and 4). Then, on day 2, it applies the best incentive (incentive 1) and updates this incentive's estimate. In so doing, it identifies that incentive 1 is not the best any more. Hence, on day 3, it applies incentive 2 (the new best incentive). We also suppose that the estimates after periods 3 and 4 are consistent with incentive 2 being the best, thus it simply applies this incentive in periods 4 and 5. Compared with the example of HAIS in Fig. 2, as HAIS eliminates the worst incentive (incentive 3) after the sampling step, it can spend more of its budget on exploring incentives 1 and 2. It applies the real best incentive (incentive 2) 6 times compared to Stepped ε-first, which only applies it 4 times. Hence, it better estimates incentive 2 and finds that this is the best incentive. In the exploitation phase, HAIS applies this incentive all the time. However, with Stepped ε-first, after the exploration phase, it applies incentive 1 twice (in period 2) before identifying and applying the real best incentive (incentive 2) with the residual budget in periods 3, 4, and 5. From this example, it can be seen that by exploring the incentives evenly without looking at their group sizes, Stepped ε-first over-explores the incentives with large group sizes and under-explores the other ones. Hence, it is easier to miss the best incentive with a small group size compared to HAIS. However, compared to ε-first, Stepped ε-first is likely to be better because it takes advantage of the residual time budget to exploit.

Fig. 17 An example of applying policy with `Stepped ε-first` where $I = 3$, $g_1 = 4$, $g_2 = 2$, $g_3 = 2$, $T = 5$ (periods), $B = £80$, $c_1 = £4$, $c_2 = £2$, $c_3 = £2$, $\epsilon_1 = 0.4$, $\epsilon_2 = 0.5$, incentive 2 is the real best incentive, and incentive 3 is the real worst incentive

References

1. Truong NV-Q, Stein S, Tran-Thanh L, Jennings NR (2018) Adaptive incentive selection for crowdsourcing contests. In: Proceedings of the 17th international conference on autonomous agents and multiagent systems (AAMAS). IFAAMAS, pp 2100–2102

2. Doan A, Ramakrishnan R, Halevy AY (2011) Crowdsourcing systems on the world-wide web. Commun ACM 54(4):86–96. https://doi.org/10.1145/1924421.1924442

3. Ghezzi A, Gabelloni D, Martini A, Natalicchio A (2018) Crowdsourcing: A review and suggestions for future research. Int J Manag Rev 20(2):343–363. https://doi.org/10.1111/ijmr.12135

4. Jain S, Deodhar SJ (2021) Social mechanisms in crowdsourcing contests: a literature review. Behaviour & Information Technology,, pp 1–35. https://doi.org/10.1080/0144929X.2021.1880638

5. Vermicelli S, Cricelli L, Grimaldi M (2021) How can crowdsourcing help tackle the COVID-19 pandemic? An explorative overview of innovative collaborative practices. R&D Management 51(2):183–194. https://doi.org/10.1111/radm.12443

6. Uzor S, Jacques JT, Dudley JJ, Kristensson PO (2021) Investigating the Accessibility of Crowdwork Tasks on Mechanical Turk. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. ACM, Yokohama Japan, pp 1–14. https://doi.org/10.1145/3411764.3445291

7. Zhen Y, Khan A, Nazir S, Huiqi Z, Alharbi A, Khan S (2021) Crowdsourcing usage, task assignment methods, and crowdsourcing platforms: A systematic literature review. Journal of Software: Evolution and Process. https://doi.org/10.1002/smr.2368

8. Callison-Burch C (2009) Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In:

9. Proceedings of the 2009 conference on empirical methods in natural language processing (EMNLP), vol 1. ACL, pp 286–295

9. Snow R, O'Connor B, Jurafsky D, Ng AY (2008) Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. In: Proceedings of the 2008 conference on empirical methods in natural language processing (EMNLP). ACL, pp 254–263

10. Vaughan JW (2018) Making better use of the crowd: how crowdsourcing can advance machine learning research. J Mach Learn Res 18(193):1–46. http://jmlr.org/papers/v18/17-234.html

11. Biswas A, Jain S, Mandal D, Narahari Y (2015) A truthful budget feasible multi-armed bandit mechanism for crowdsourcing time critical tasks. In: Proceedings of the 14th international conference on autonomous agents and multiagent systems (AAMAS). IFAAMAS, pp 1101–1109

12. Itoh A, Matsubara S (2016) Designing incentives for crowdsourced tasks via multi-armed bandits. In: IEEE international conference on agents (ICA). IEEE, pp 70–73

13. Itoh Y, Matsubara S (2021) Adaptive Budget Allocation for Cooperative Task Solving in Crowdsourcing. In: 2021 IEEE international conference on big data (Big Data). IEEE, Orlando, FL, USA, pp 3525–3533, https://doi.org/10.1109/BigData52589.2021.9671713. 00000

14. Jain S, Ghalme G, Bhat S, Gujar S, Narahari Y (2016) A deterministic MAB mechanism for crowdsourcing with logarithmic regret and immediate payments. In: Proceedings of the 15th international conference on autonomous agents and multiagent systems (AAMAS). IFAAMAS, pp 86–94

15. Kara YE, Genc G, Aran O, Akarun L (2018) Actively estimating crowd annotation consensus. J Artif Intell Res 61:363–405. https://doi.org/10.1613/jair.5727

16. Luo Y, Jennings NR (2021) A budget-limited mechanism for category-aware crowdsourcing of multiple-choice tasks. Artif Intell 299:103538. https://doi.org/10.1016/j.artint.2021.103538, https://linkinghub.elsevier.com/retrieve/pii/S0004370221000898

17. Muldoon C, O'Grady MJ, O'Hare GMP (2018) A survey of incentive engineering for crowdsourcing. The Knowledge Engineering Review 33:e2. https://doi.org/10.1017/S0269888918000061

18. Sen S, Ridgway A, Ripley M (2015) Adaptive budgeted bandit algorithms for trust development in a supply-chain. In: Proceedings of the 14th international conference on autonomous agents and multiagent systems (AAMAS). IFAAMAS, pp 137–144

19. Tran-Thanh L, Huynh TD, Rosenfeld A, Ramchurn SD, Jennings NR (2014) BudgetFix: Budget limited crowdsourcing for

interdependent task allocation with quality guarantees. In: Proceedings of the 13th international conference on autonomous agents and multiagent systems (AAMAS). IFAAMAS, pp 477–484

20. Truong NV-Q, Stein S, Tran-Thanh L, Jennings NR (2019) What prize is right? How to learn the optimal structure for crowdsourcing contests. In: Proceedings of the 16th pacific rim international conference on artificial intelligence (PRICAI). Springer International Publishing, pp 85–97

21. Venanzi M, Guiver J, Kohli P, Jennings NR (2016) Time-sensitive bayesian information aggregation for crowdsourcing systems. J Artif Intell Res 56:517–545. https://doi.org/10.1613/jair.5175

22. Simula H (2013) The rise and fall of crowdsourcing? In: Proceeding of the 46th Hawaii international conference on system sciences (HICSS). IEEE, pp 2783–2791. https://doi.org/10.1109/HICSS.2013.537

23. Mason W, Watts DJ (2010) Financial incentives and the "performance of crowds". ACM SigKDD Explorations Newsletter 11(2):100–108

24. Harris C (2011) You're hired! An examination of crowdsourcing incentive models in human resource tasks. In: Proceedings of the workshop on crowdsourcing for search and data mining at the Fourth ACM International Conference on Web Search and Data Mining (WSDM). ACM, pp 15–18

25. Yin M, Chen Y (2015) Bonus or not? Learn to reward in crowdsourcing. In: Proceedings of the 24th international joint conference on artificial intelligence (IJCAI). AAAI Press, pp 201–207

26. Frey BS, Jegen R (2001) Motivation crowding theory. J Econ Surv 15(5):589–611. https://doi.org/10.1111/1467-6419.00150

27. Heyman J, Ariely D (2004) Effort for payment: A tale of two markets. Psychol Sci 15(11):787–793

28. Zheng H, Li D, Hou W (2011) Task design, motivation, and participation in crowdsourcing contests. Int J Electron Commer 15(4):57–88. https://doi.org/10.2753/JEC1086-4415150402

29. Ramchurn SD, Huynh TD, Venanzi M, Shi B (2013) Collabmap: Crowdsourcing maps for emergency planning. In: Proceedings of the 5th annual ACM Web science conference (WebSci). ACM, pp 326–335

30. Tran-Thanh L, Chapman A, Munoz De Cote Flores Luna JE, Rogers A, Jennings NR (2010) Epsilon–first policies for budget–limited multi-armed bandits. In: Proceedings of the 24th AAAI conference on artificial intelligence. AAAI Press, pp 1211–1216

31. Tran-Thanh L, Chapman AC, Rogers A, Jennings NR (2012) Knapsack based optimal policies for budget-limited multi-armed bandits. In: Proceedings of the 26th AAAI conference on artificial intelligence. AAAI Press, pp 1134–1140

32. Badanidiyuru A, Kleinberg R, Slivkins A (2018) Bandits with knapsacks. J ACM 65(3):1–55. https://doi.org/10.1145/3164539

33. Archak N, Sundararajan A (2009) Optimal design of crowdsourcing contests. In: Proceedings of the 13th international conference on information systems (ICIS). AIS, pp 1–16

34. Cavallo R, Jain S (2012) Efficient crowdsourcing contests. In: Proceedings of the 11th international conference on autonomous agents and multiagent systems (AAMAS), vol 2. IFAAMAS, pp 677–686

35. Cavallo R, Jain S (2013) Winner-take-all crowdsourcing contests with stochastic production. In: Proceedings of the 1st AAAI conference on human computation and crowdsourcing (HCOMP). AAAI Press, pp 34–41

36. Chawla S, Hartline JD, Sivan B (2012) Optimal crowdsourcing contests. In: Proceedings of the 23d annual ACM-SIAM symposium on discrete algorithms (SODA). SIAM, pp 856–868

37. DiPalantino D, Vojnovic M (2009) Crowdsourcing and all-pay auctions. In: EC '09 Proceedings of the 10th ACM conference on Electronic commerce. ACM, pp 119–128

38. Korpeoglu CG, Körpeoğlu E, Tunç S (2021) Optimal duration of innovation contests. Manufacturing & Service Operations Management 23(3):657–675. https://doi.org/10.1287/msom.2020.0935

39. Luo T, Kanhere SS, Tan H-P, Wu F, Wu H (2015) Crowdsourcing with tullock contests: A new perspective. In: IEEE conference on computer communications (INFOCOM). IEEE, pp 2515–2523

40. Luo T, Das SK, Tan HP, Xia L (2016) Incentive mechanism design for crowdsourcing: An all-pay auction approach. ACM Trans Intell Syst Technol 7(3):1–26. https://doi.org/10.1145/2837029

41. Moldovanu B, Sela A (2001) The optimal allocation of prizes in contests. Am Econ Rev 91(3):542–558

42. Gneezy U, Rustichini A (2000) Pay enough or don't pay at all. Q J Econ 115(3):791–810

43. Rogstadius J, Kostakos V, Kittur A, Smus B, Laredo J, Vukovic M (2011) An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In: Proceedings of the 5th international AAAI conference on Weblogs and Social Media (ICWSM). AAAI Press, pp 321–328

44. Auer P, Cesa-Bianchi N, Freund Y, Schapire RE (2002) The nonstochastic multiarmed bandit problem. SIAM J Comput 32(1):48–77. https://doi.org/10.1137/S0097539701398375

45. Thompson WR (1933) On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika 25(3/4):285–294. https://doi.org/10.2307/2332286

46. Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. Mach Learn 47(2-3):235–256

47. Ho C-J, Slivkins A, Vaughan JW (2016) Adaptive contract design for crowdsourcing markets: Bandit algorithms for repeated principal-agent problems. J Artif Intell Res 55:317–359. https://doi.org/10.1613/jair.4940

48. Kaufmann N, Schulze T, Veit D (2011) More than fun and money. Worker motivation in crowdsourcing – A study on Mechanical Turk. In: AMCIS '11 Proceedings of the 7th Americas Conference on Information Systems, vol 11. AIS, pp 1–11

49. Araujo RM (2013) 99designs: An analysis of creative competition in crowdsourced design. In: Proceedings of the 1st AAAI conference on human computation and crowdsourcing (HCOMP). AAAI Press, pp 17–24

50. Feyisetan O, Simperl E (2019) Beyond monetary incentives: experiments in paid microtask contests. ACM Transactions on Social Computing 2(2):1–31 (en). https://doi.org/10.1145/3321700 , http://dl.acm.org/citation.cfm?doid=3340675.3321700

51. Talbi E-G (2009) Metaheuristics: from design to implementation. John Wiley & Sons, Hoboken, N.J

52. Hoeffding W (1963) Probability inequalities for sums of bounded random variables. J Am Stat Assoc 58(301):13–30. https://doi.org/10.2307/2282952

53. Sutton RS, Barto AG (2018) Reinforcement Learning: An Introduction, 2nd edn. Adaptive Computation and Machine Learning Series, The MIT Press

54. Yang J, Adamic LA, Ackerman MS (2008) Crowdsourcing and knowledge sharing: Strategic user behavior on Taskcn. In: Proceedings of the 9th ACM conference on electronic commerce (EC). ACM, pp 246–255

55. Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems 25 (NIPS), vol 2. Curran Associates, Inc., Nevada, USA, pp 2951–2959. 00000

56. Victoria AH, Maragatham G (2021) Automatic tuning of hyperparameters using Bayesian optimization. Evolving Systems 12(1):217–223. https://doi.org/10.1007/s12530-020-09345-2

57. Mandel T, Liu Y-E, Brunskill E, Popovic Z (2015) The queue method: Handling delay, heuristics, prior data, and evaluation in bandits. In: Proceedings of the 29th AAAI conference on artificial intelligence. AAAI Press, pp 2849–2856

Affiliations

Nhat Van-Quoc Truong[1] · **Le Cong Dinh[1]** · **Sebastian Stein[1]** · **Long Tran-Thanh[2]** · **Nicholas R. Jennings[3]**

Le Cong Dinh
l.c.dinh@soton.ac.uk

Sebastian Stein
s.stein@soton.ac.uk

Long Tran-Thanh
long.tran-thanh@warwick.ac.uk

Nicholas R. Jennings
n.r.jennings@lboro.ac.uk

[1] Electronics and Computer Science, University of Southampton, Southampton, UK

[2] Department of Computer Science, University of Warwick, Coventry, UK

[3] Loughborough University, Loughborough, UK