What Prize is Right? How to Learn the Optimal Structure for Crowdsourcing Contests

Nhat Van-Quoc Truong[1], Sebastian Stein[1], Long Tran-Thanh[1], and Nicholas R. Jennings[2]

[1] Electronics and Computer Science, University of Southampton, UK
{n.truong,s.stein,l.tran-thanh}@soton.ac.uk
[2] Department of Computing, Department of Electrical and Electronic Engineering, Imperial College, London, UK
n.jennings@imperial.ac.uk

Abstract. In crowdsourcing, one effective method for encouraging 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. They could vary in their structure (e.g., performance evaluation and the number of prizes) and parameters (e.g., the maximum number of participants and the amount of prize money). Additionally, with a given budget and a time limit, choosing incentives (i.e., contest structures with specific parameter values) that maximise the overall utility is not trivial, as their respective effectiveness in a specific project is usually unknown a priori. Thus, in this paper, we propose a novel algorithm, *BOIS* (Bayesian-optimisation-based incentive selection), to learn the optimal structure and tune its parameters effectively. In detail, the learning and tuning problems are solved simultaneously by using online learning in combination with Bayesian optimisation. The results of our extensive simulations show that the performance of our algorithm is up to 85% of the optimal and up to 63% better than state-of-the-art benchmarks.

Keywords: Incentive · Crowdsourcing · Bayesian Optimisation.

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 (Ghezzi et al., 2018; Doan et al., 2011). However, the success of crowdsourcing projects relies critically on a crowd to contribute (Simula, 2013; Doan et al., 2011). Given this, contests[3] have been shown to be an effective

[3] 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 as contests for the purpose of this paper.

approach in these projects, as they are effective and cheap. In particular, 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 paying for performance (Mason and Watts, 2010) or using bonuses (Yin and Chen, 2015). Indeed, they have to pay only 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. 99designs (www.99designs.com), TopCoder (www.topcoder.com), and Taskcn (www.taskcn.com) are some well-known crowdsourcing platforms that use contests to attract participants.

Much work has taken a game-theoretic approach to investigate the optimal (or efficient) design of contests in general and crowdsourcing contests in particular. It tries to answer the questions of how to distribute the prizes (number of prizes and their values) in contests (Luo et al., 2015; Cavallo and Jain, 2012; Moldovanu and Sela, 2001). Yet, applying this body of research in building efficient contests for real-world crowdsourcing projects is still challenging. This is because these studies assume rational participants, whereas real participants in crowdsourcing might be partly rational or irrational, as they might lack information, knowledge, or time. Also, these studies do not consider other factors related to the participants' intrinsic motivation that might affect their behaviour such as the project purpose (e.g., collecting data for scientific studies, for government agencies or for companies) or the task nature (e.g., interesting or boring) (Rogstadius et al., 2011; Frey and Jegen, 2001).

Furthermore, currently on many crowdsourcing platforms such as Amazon Mechanical Turk (www.mturk.com) and Figure Eight (www.figure-eight.com), the requesters can create tasks and get the submissions in an autonomous manner using programmable Application Programming Interfaces (APIs). This makes it possible to build autonomous agents to monitor and adaptively switch contest structures (e.g., performance evaluation and the number of prizes) and parameters (e.g., the maximum number of participants and the amount of prize money) when appropriate. We refer to a contest structure with specific values of the parameters as an *incentive*[4]. Indeed, it is inconvenient or almost impossible in many cases to switch between incentives manually to identify the best one.

Therefore, another direction for dealing with the incentive problem is to design incentives that are likely to be effective based on previous studies and then empirically select the most effective one. In detail, the above-mentioned studies can be used to design several contest structures with specific ranges of their parameters which are referred to as *candidate incentives*. Then, based on the proposed candidates, an adaptive approach could be used to identify the most effective candidate efficiently. Hence, finding an appropriate way for an autonomous agent (i.e., a computer programme) to select an effective incentive

[4] 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 pay for performance or bonuses. Thus, to keep the problem general, we use the term "incentives" instead of "contest structures".

in a crowdsourcing project is a key problem. We refer to this as the *incentive selection problem* (ISP) (Truong et al., 2018).

To identify the most effective incentive to utilise (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 (MABs) are a promising approach for the ISP. Specifically, they model the problem as a machine with N arms (corresponding to N incentives), pulling an arm (offering the corresponding incentive to a group of participants) incurs a fixed cost (attached to the arm) and delivers a random utility (e.g., the number of tasks completed) drawn from an unknown distribution. The objective in an MAB problem is to find a policy that maximises the total utility within a given budget (e.g., £500) before a deadline (e.g., in the next two weeks).

A number of studies about budgeted MABs have been conducted, such as Badanidiyuru et al. (2018), Ho et al. (2016), and Tran-Thanh et al. (2010). But these studies cannot be applied directly to the ISP as they cannot deal with the tuning problem (i.e., choosing appropriate parameter values for a contest structure) effectively. This is because they do not take advantage of the possible correlations between the arms (i.e., the incentives in a contest structure). Many-armed bandits work well with many or even an infinite number of arms (Li and Xia, 2017; Trovo et al., 2016; Bubeck et al., 2011). Yet, none of them can be used to solve the ISP. Actually, they do not consider all important characteristics of the ISP, such as the budget constraints, multidimensional structure of the incentives (i.e., a contest structure has a certain number of parameters), correlations between the arms, and the group-based nature of the arm. Bayesian optimisation (BO) is shown to be an efficient alternative (Snoek et al., 2012). Indeed, BO is designed to find the global optima of functions in as few steps (i.e., function evaluations) as possible. This fits the ISP as applying an incentive incurs a cost. Also, as BO incorporates prior beliefs, if we have some prior knowledge about user performance in the current crowsourcing project, BO can make use of this to find the global optimum more quickly.

Therefore, in this paper we combine the two (online learning with MABs and tuning with BO) to deal with the ISP. By so doing, we decouple a complicated problem (with both learning the best structure and tuning its parameters) into two simple problems and deal with these in a learning process). 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:

(1) We formalise the ISP and then introduce BOIS, a novel algorithm to solve the ISP effectively by combining an MAB approach to learn the contest structures and BO to tune the parameters of the structures.
(2) We empirically demonstrate that BOIS is generally more effective compared to the state-of-the-art approaches in an extensive series of simulations.

2 The Incentive Selection Problem

Suppose a requester wants to run a crowdsourcing project. The objective is typically to maximise the requester's overall utility with a given financial budget B and time budget T. We can include task quantity, task quality, task completion time, or some subset of them in the utility function. For example, Yin and Chen (2015) consider the quantity and quality of the tasks. To achieve this objective, the requester spends the available budget on providing incentives to encourage participants (referred to as *users*) to perform tasks. For a better presentation, we group the incentives with the same structure in a cluster, which is referred to as incentive cluster (or *cluster* for short). We assume that there are correlations between different incentives in a cluster. Figure 1 shows possible correlations between the incentives in a cluster. Specifically, Figure 1a shows the effectiveness of the incentives, measured by utility per cost unit[5], in a cluster when there is only one parameter (group size). This figure depicts that the utility initially increases with increasing group size. However, when it is larger than 20, the effectiveness starts decreasing. Figure 1b shows another example in a cluster with two parameters, the group size and the amount of prize money for the best user. We are interested in finding an effective means of selecting the incentives (i.e., exploring their effectiveness and exploiting the most effective one) in order to maximise the requester's overall utility. This is referred to as the ISP.

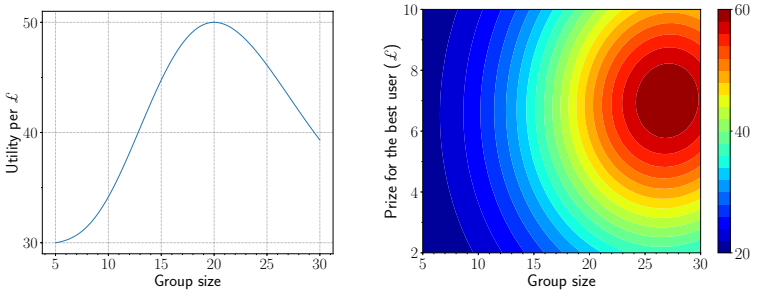
Formally, let C denote the number of clusters that are being considered in a crowdsourcing project. Cluster i (or C_i for short) has K_i parameters. An incentive a in C_i corresponds to a structure vector $v_a = (v_a^{(1)}, \ldots, v_a^{(K_i)})$, where $v_a^{(k)}$ is the value corresponding to the k^{th} parameter and $v_a^{(k)} \in [\mathrm{v}_{min,i}^{(k)}, \mathrm{v}_{max,i}^{(k)}]$ ($\mathrm{v}_{min,i}^{(k)}, \mathrm{v}_{max,i}^{(k)} \in \mathbb{R}$). Let g_a be the group size of a and c_a denote the cost of applying incentive a once. The expected utility of a is μ_a which is unknown a priori. Let $\mathcal{N} = \{n_a^{(t)} \mid t = 1, \ldots, T; a \in C_i; i = 1, \ldots, C\}$ denote a policy, where $n_a^{(t)}$ is the number of times incentive a is applied in period t, i.e., incentive a is offered to $n_a^{(t)}$ different groups. Let $u_a^{(t)}$ be the total utility of applying this incentive $n_a^{(t)}$ times in period t. The objective is to find a policy that maximises the overall utility:

$$\max \sum_{t=1}^{T} \sum_{i=1}^{C} \sum_{a \in C_i} u_a^{(t)} \quad \text{subject to} \quad \sum_{t=1}^{T} \sum_{i=1}^{C} \sum_{a \in C_i} n_a^{(t)} c_a \leq B. \tag{1}$$

3 The BOIS Algorithm

In this section, we introduce BOIS (which stands for BO-based Incentive Selection), a novel algorithm for the ISP. However, we first describe how the algorithm and the benchmarks measure the effectiveness of the incentives (Subsection 3.1). We then give an overview of the algorithm (Subsection 3.2). Finally, we detail how

[5] The measurement of an incentive's effectiveness will be discussed in Subsection 3.1.



(a) Cluster 1 (b) Cluster 2

Fig. 1. Illustrative examples of correlations between the incentives in a cluster when it has one (a) and two (b) parameters.

BOIS splits the learning and tuning process into steps and how it acts in these steps (Subsections 3.3–3.5).

3.1 Measuring the Effectiveness of the Incentives

To measure the effectiveness of the incentives, we use the utility-cost ratio[6], as it reflects the average utility per cost unit. The effectiveness of incentive a is defined as $\delta_a = \mu_a / c_a$. However, as the real effectiveness of the incentives are unknown in advance, we have to estimate them. Right after period t, the estimate of incentive a's effectiveness is:

$$d_a^{(t)} = \hat{\mu}_a^{(t)} / c_a, \tag{2}$$

where $\hat{\mu}_a^{(t)} = \left(1/m_a^{(t)}\right) \sum_{\tau=1}^{t} u_a^{(\tau)}$ is the current estimate of incentive a's expected utility $\left(m_a^{(t)}\right.$ is the number of times incentive a has been applied until the end of period $\left.t\right)$. To keep the presentation simple, we use *the best incentive* to denote the incentive with the highest estimate, as opposed to *the real best incentive*.

3.2 Algorithm Overview

The idea of BOIS is using an MAB approach to deal with the learning problem (i.e., identifying the best cluster) and using BO[7] with Gaussian processes to tackle the tuning problem (i.e., finding the optimal values of the parameters of a cluster). In more detail, in each period (except the first one), it selects the incentive whose value of the acquisition function corresponding to this incentive is the largest compared to those of the other incentives in all clusters. Note that in BO, acquisition functions are to propose the next sampling incentive in the search space. We have tried several acquisition functions such as expected improvement, maximum probability of improvement, and upper confidence bound

[6] This ratio is called "density" in Tran-Thanh et al. (2010).
[7] See Snoek et al. (2012) for more information about the method.

(UCB). However, we chose the UCB (which is the upper confidence bound of the estimate of the incentive's effectiveness) as it is the most effective.

The general idea of tuning parameter values of a contest structure (i.e., finding the real best incentive in a cluster) using BO with Gaussian processes is the following. In each period, based on the incentives sampled in the previous periods, BOIS estimates the mean utilities of the incentives in the cluster using Gaussian process regression (GPR). Then, it calculates the UCBs of the incentives. After that, the incentive with the highest UCB will be the candidate to be applied next in the cluster. BOIS will then choose the candidate incentive in the cluster which has the highest UCB to be applied in that period. In order for the algorithm to use BO, it must have initial estimates of the incentives in each cluster. Therefore, in the first period (i.e., period 1), it samples several incentives, in order to obtain good estimates of the incentives. This step is referred to as the *sampling step*. Then, in each of the next periods (except the last one), it applies the most promising incentive (a), i.e., the incentive with the largest UCB. After that, it updates the UCBs of the incentives in the same cluster (i.e., C_i if $a \in C_i$). We refer to this step as the *stepped exploitation step*. Finally, in the last period it applies the best incentive with the remaining budget. This step is called the *pure exploitation step*, as it simply exploits the best incentive after exploring in the previous periods.

Regarding the UCBs, to select an incentive in a period $t+1$, at the end of the previous period (t), BOIS uses GPR to estimate the mean utilities of all incentives in each cluster. The results of the estimation are $\hat{\mu}_a^{(t)}$ and $\hat{\sigma}_a^{(t)}$ $\forall a \in C_i; i = 1, \ldots, C$. Then, it calculates the potential effectiveness of all incentives:

$$d_a^{*(t)} = \frac{1}{c_a}\left(\hat{\mu}_a^{(t)} + z^{(t)}\frac{\hat{\sigma}_a^{(t)}}{\sqrt{m_a^{(t)} g_a}}\right). \tag{3}$$

In Equation 3, $z^{(t)} = Z\big(1 - \frac{t-1}{T-2}\big)$, where Z is the critical value (e.g., 1.96) corresponding to the initial confidence level (e.g., 95%) of the estimates. In more detail, as in the first periods we are not confident about the estimates of the incentives, the confidence intervals should be large to make sure that the algorithm does not leave out the real best incentive. That means at first, it is better to focus on exploration. Then in the next periods, the intervals should become smaller gradually. By so doing, it not only solves the learning and tuning problems simultaneously, but also it performs a smooth transition from exploration to exploitation. Literally, the first period ($t = 1$), $z^{(t)} = Z$ means that it focuses more on exploration. Then, its value gradually decreases as time goes by. And finally, when $t = T$, $z^{(t)} = 0$ means that it focuses only on exploitation. Additionally, the denominator, $\sqrt{m_a^{(t)} g_a}$, signifies that the exploration level is inversely propotional to the number of sampled users.

In the next subsections, details of the steps will be discussed. The explanations will be linked to the corresponding parts of the pseudocode of BOIS shown in Algorithm 1.

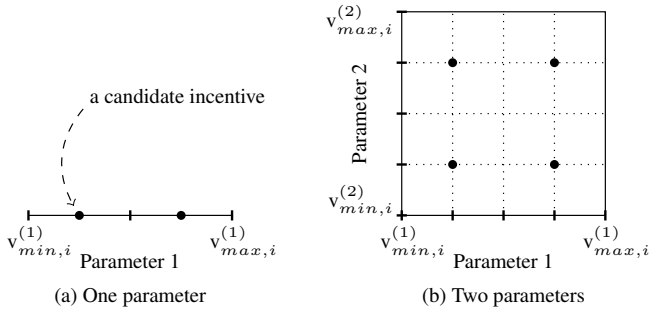


Fig. 2. An illustration of candidate incentives in a cluster in the Sampling step when the cluster has one (a) and two (b) parameters

3.3 The Sampling Step

As mentioned above, the purpose of this step (Lines 2–10) is to obtain initial estimates of the incentives in each cluster, which are then used for the regression in the next step. BOIS uses the miniMax distance design (Johnson et al., 1990) to sample the incentives in each cluster to ensure that all other incentives in the cluster are not too far from the sampled ones. An illustration of this space-filling design is shown in Figure 2. In more detail, for the k^{th} parameter of cluster i, BOIS chooses two values, one in the first quarter and the other in the third quarter of its range, i.e., $\mathrm{v}^{(k)}_{min,i} + 0.25\Delta^{(k)}_i$ and $\mathrm{v}^{(k)}_{min,i} + 0.75\Delta^{(k)}_i$, where $\Delta^{(k)}_i = v^{(k)}_{max,i} - v^{(k)}_{min,i}$ (Figure 2a). From these values, we have a set of 2^{K_i} candidate incentives to be sampled. Figure 2b shows four candidate incentives in a cluster which has two parameters.

One issue is that the financial budget is limited and we also want to spend it on further exploration and exploitation. So, BOIS only uses $\epsilon_1 B$ (e.g., $0.2B$) for sampling. This amount might not be enough to sample all the above-mentioned 2^{K_i} candidate incentives ($\forall i = 1, \ldots, C$). Therefore, BOIS simply iterates over the clusters (Line 4) and at each cluster it chooses a random (without repetition) incentive from this set. This is conducted by the NextSample() function (Line 5). Once an incentive is chosen, it will be applied several times so that it has about $\mathbf{U}_1$ (e.g., 20) sampled users, which is calculated by rounding the division $\mathbf{U}_1/g_a$ to the nearest integer (Lines 8–9). By so doing, it guarantees to have enough sampled users if the group size of the incentive is small (e.g., 2). Note that $\lfloor b_1/c_a \rfloor$ in Line 8 is to guarantee the budget being used in this step does not exceed $\epsilon_1 B$. BOIS stops sampling when the budget for sampling is exceeded (Lines 6–7).

3.4 The Stepped Exploitation Step

At first, BOIS sets the budget for stepped exploitation, a specific portion of the residual budget which is identified by ϵ_2, e.g., 0.5 (Line 11). Then, in each period (t) before the deadline, it will choose the incentive (a) with the highest potential effectiveness (Line 13). The incentives are chosen based on their UCBs which

Algorithm 1 The BOIS Algorithm

Input: $B, T, C,$ and $K_i \ \ \forall i = 1, \ldots, C$
Predefined parameters: $\epsilon_1, \epsilon_2, \mathbf{U}_1, D_{min}$, and Z
Output: $u, \mathcal{N} = \{n_a^{(t)} \mid t = 1, \ldots, T; a \in C_i; i = 1, \ldots, C\}$
Note: ApplyIncentive(a, n) is to apply incentive a n times and return the total utility.

01: $b \leftarrow B$; ▷ overall residual budget
Sampling:
02: $b_1 \leftarrow \epsilon_1 B$; ▷ residual budget for sampling
03: **while** true **do**
04: **for** $i = 1 \rightarrow C$ **do**
05: $a \leftarrow$ NextSample(C_i);
06: **if** $b_1 < c_a$ **then** ▷ sampling budget is exceeded
07: Stop the for and while loops;
08: $n_a^{(1)} \leftarrow \max\left\{1, \min\{[\mathbf{U}_1/g_a], \lfloor b_1/c_a \rfloor\}\right\}$;
09: $u_a^{(1)} \leftarrow \text{ApplyIncentive}\left(a, n_a^{(1)}\right)$; $b_1 \leftarrow b_1 - n_a^{(1)} c_a$; $b \leftarrow b - n_a^{(1)} c_a$;
10: UpdateEstimates$(C_i, 1, Z) \ \ \forall i = 1, \ldots, C$;
Stepped Exploitation:
11: $b_2 \leftarrow \epsilon_2 b$; ▷ residual budget for exploration
12: **for** $t = 2 \rightarrow T - 1$ **do**
13: $a \leftarrow \text{argmax}_{a' \in C_i;\ i=1,\ldots,C}\{d_{a'}^{*(t-1)}\}$;
14: **if** $d_a^{*(t-1)} < D_{min}$ **then** ▷ a is too bad
15: $i \leftarrow$ a random cluster; $a \leftarrow$ a random incentive in C_i;
16: **if** $b_2 < c_a$ **then** ▷ budget for exploration is exceeded
17: Stop the for loop;
18: $n_a^{(t)} \leftarrow \max\left\{1, \min\{[\mathbf{U}_1/g_a], \lfloor b_2/c_a \rfloor\}\right\}$;
19: $u_a^{(t)} \leftarrow \text{ApplyIncentive}\left(a, n_a^{(t)}\right)$; $b_2 \leftarrow b_2 - n_a^{(t)} c_a$; $b \leftarrow b - n_a^{(t)} c_a$;
20: UpdateEstimates$(C_i, t, Z,)$;
Pure Expl.:
21: $a \leftarrow \text{argmax}_{a' \in C_i;\ i=1,\ldots,C}\{d_{a'}^{*(T-1)}\}$;
22: $n_a^{(T)} \leftarrow \max\left\{1, \lfloor b/c_a \rfloor\right\}$;
23: $u_a^{(T)} \leftarrow \text{ApplyIncentive}\left(a, n_a^{(T)}\right)$;
24: $u \leftarrow \sum_{t=1}^{T} \sum_{i=1}^{C} \sum_{a \in C_i} u_a^{(t)}$; ▷ overall utility
25: **return** $u, \mathcal{N}$;

contain both the estimates of the incentives' effectiveness so far and the certainty of the estimates. Thus, this step can be considered as both exploiting (choosing the incentives whose estimates are high) and exploring (choosing the incentives whose potential to be the real best one are high).

In some cases, the potential effectiveness of this incentive ($d_a^{*(t-1)}$) can be very low since the sampled incentives so far in this cluster (C_i) had very low utilities. To prevent it from falling into the trap of exploring ineffective incentives, if $d_a^{*(t-1)}$ is less than some lower bound (D_{min}), BOIS will randomly choose another incentive (Lines 14–15). It is not difficult to determine a value for D_{min}. For example, if the utility is measured by the number of tasks completed and we expect an acceptable incentive to have about 20 completed tasks per £, then we can set D_{min} to this value or even 10 if we are not quite sure about this number.

Algorithm 2 The UpdateEstimates() Function

Input: C_i, t and Z
Output: C_i with updated $d_a^{*(t)} \quad \forall a \in C_i$
01: Use Gaussian process regression to estimate $\hat{\mu}_a^{(t)}$ and $\hat{\sigma}_a^{(t)} \quad \forall a \in C_i$;
02: Calculate $d_a^{*(t)}$ based on Equation 3 $\quad \forall a \in C_i$;

Yet, it should be larger than the possible minimum number of tasks, e.g., 0. As in the sampling step, after having an incentive, BOIS will apply the incentive several times so that it obtains about $\mathbf{U}_1$ sampled users (Lines 18–19). This step stops when the budget for exploration is exceeded (Lines 16–17).

3.5 The Pure Exploitation Step

This step (Lines 21–23) simply applies the best incentive with the residual budget. Indeed, from Equation 3 we can see that in this period the factor $z^{(T)}$ is zero. That means it does not explore anymore but totally exploits the incentive with the highest estimate of the effectiveness.

4 Experimental Evaluation

To systematically evaluate the performance of BOIS, we run simulations in a wide range of settings. It would be infeasible to undertake this evaluation in a real crowdsourcing project as we have to deploy the project multiple times with different financial budgets, time budgets, and numbers of clusters, as well as different values of the parameters of each cluster. Even then, we could not guarantee we have explored the main cases in a comprehensive fashion. In the following, we present the benchmarks (Subsection 4.1), the experimental settings (Subsection 4.2), and then discuss the corresponding results (Subsection 4.3).

4.1 Benchmarks

As the state-of-the-art algorithms are not specifically designed to deal with choosing the best cluster together with tuning its parameter values, we make a number of modifications for them to perform well with the ISP.

(1) **ϵ-first**: This algorithm spends ϵB (where ϵ is specified in advance, e.g., 0.1) in the first period to explore by sequentially applying a random incentive in each cluster until this budget is exceeded (Tran-Thanh et al., 2010). With a chosen incentive a, it applies this incentive $\max\{1, [\mathbf{U_1}/c_a]\}$ times to obtain about $\mathbf{U}_1$ (e.g., 20) sampled users. In the second period, it uses GPR to estimate the best incentive. Then it spends the subsequent period purely exploiting the best incentive explored in the first period with the remaining budget, i.e., $(1-\epsilon)B$.

(2) **Decaying ϵ-greedy** (or ϵ-greedy for short): It spreads the budget B over T periods. In each period, with the given budget, it applies the best incentive with probability $(1-\epsilon)$ and a random incentive in a random cluster with probability ϵ, where $\epsilon = (T-t)/(T-1)$. It totally explores when $t = 1$ (i.e., $\epsilon = 1$). When t increases, ϵ gradually decreases. And when $t = T$, it completely exploits the best incentive (i.e., $\epsilon = 0$). At the end of period t $(1 < t < T)$, it uses GPR to estimate the best incentive for period $t+1$.
(3) **Random**: It spreads the budget B over T periods. Then in each period, it applies a random incentive in a random cluster with the given budget.
(4) **Optimal Solution**: It simply applies the real best incentive all the time. To have this optimality, we have to know the values μ_a $(\forall a \in C_i; \forall i = 1, \ldots, C)$ in advance, which is typically impossible in practice. Thus, this approach represents an upper bound of what any algorithm could achieve.

4.2 Simulation Settings

To evaluate the performance of the algorithms, we run simulations in three different settings where the independent variables are time budget, financial budget, and number of clusters. In the simulations of each setting, the related quantities, i.e., utility, B, T, C, group size, and the amount of prize money for the best user (except the corresponding independent variable) are drawn uniformly from specific ranges. The ranges are chosen to represent realistic settings in real crowdsourcing projects. Specifically, C is generated randomly from 1 to 10. The group sizes (g_a) are from 1 to 50. The amount of prize money for the best user is between £1 and £25. T is between 2 and 30. And B is from 10 to 200 times the round cost. Here, round cost is the cost of applying all the clusters, where in each cluster the incentive which the highest cost is applied once. This is to guarantee B is not too small compared to the generated values of C and g_a, so that we can carry out a meaningful performance comparison.

For each value of the independent variables, we run 2,000 simulations to achieve statistically significant results at the 99% confidence level. Error bars of the line graphs in Figure 3 represent the confidence intervals. We run the algorithms with different values of the predefined parameters and then choose appropriate values for the parameters. For instance, with ϵ 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_1 and ϵ_2 of BOIS. 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 Subsection 4.3 we only present the results on the simulations with the following values of the algorithms' predefined parameters. With BOIS, $\epsilon_1 = 0.1$, $\epsilon_2 = 0.5$, $\mathbf{U}_1 = 20$, and $Z = 1.96$. With ϵ-first, $\epsilon = 0.1$ and $\mathbf{U}_1 = 20$. In the simulations, we assume that the performance of a group (i.e., the total utility of all users in the group) is linearly proportional to the group size. This means the more users there are in a group, the better the performance of the whole group. This assumption is based on an empirical study conducted by Araujo (2013).

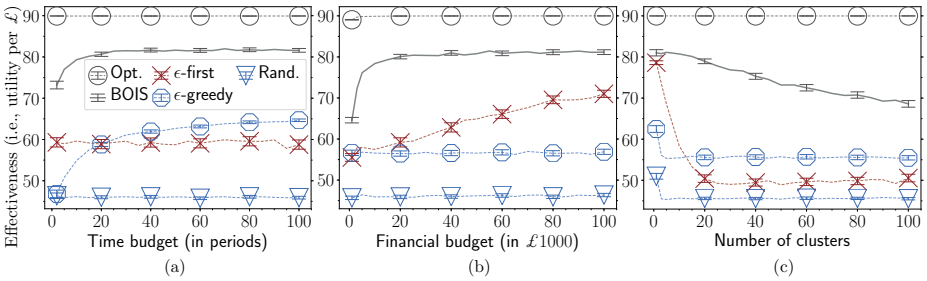


Fig. 3. Results of the simulations

4.3 Results

In general, BOIS performs best in most cases (Figure 3). With a looser deadline, the algorithm performs better, especially when T is greater than 15 (Figure 3a). This is mainly because of the miniMax space-filling design and the BO. Specifically, if $T = 2$ (i.e., no exploration) the performance of BOIS is good enough (which is a utility of about 70 per £). And if $T = 15$, its performance increases clearly (up to about 79 per £). Note that the time budget is used to learn all the clusters. This confirms that BO can quickly approach a global optimum (i.e., the real best incentive). ϵ-greedy also performs better with a larger T, since it has more time to explore. Yet, its performance is far below that of BOIS. Whereas, different values of T does not affect the performance of ϵ-first as it always uses two periods. Nonetheless, with a larger financial budget, ϵ-first performs better, as there is more budget for exploration (Figure 3b). As the way it explores is inflexible (i.e., always ϵB), when B is small, the budget for exploration is not enough, so that the GPR conducted in the second period does not have enough samples to identify one of the best incentives.

Figure 3b suggests that B should be large enough (e.g., at least £5000 as in the simulations) for BOIS to achieve a good performance. A larger B helps improve its performance slightly. Actually, it needs enough budget to sample all 2^{K_i} candidate incentives $\forall i = 1, \ldots, C$. And with a larger B, the amount of the added budget will be used for exploiting. In Figure 3c, the performance of BOIS drops significantly when C becomes larger. This is easy to understand, as with a fixed B and a larger C, $\epsilon_1 B$ is not enough to sample all the candidate incentives in all clusters. Similarly, ϵ-first's performance drops more quickly than that of BOIS. The reason is that it does not make use of the time budget to conduct further exploration. Regarding the number of parameters, as BOIS does not scale well to settings with large values of K_i, we only ran experiments with K_i=2, 3, and 4. These results have a similar trend as in Figure 3c, i.e., that BOIS performs well when $K_i = 2$ (a utility of about 82 per £). Then, its performance drops down to about 69 when $K_i = 3$ and 62 when $K_i = 4$. Also, even when $K_i = 4$, the time to run the algorithm (the whole episode, i.e., $t = 1, \ldots, T$) is less than one minute, which is acceptable in practice.

The results suggest several guidlines for using BOIS effectively in practice. First, both T and B should be large enough and a larger T has more effect

on the algorithm. Second, C and K_i ($\forall i = 1, \ldots, C$) should be small. If there are many (e.g., 15) candidate clusters to choose from, it is better to continue using related studies from psychology, sociology, or computer science to filter out clusters which are not actually promising. A similar process should be done with the parameters.

5 Conclusions and Future Work

We have discussed the incentive selection problem (ISP) and introduced an algorithm (BOIS) to solve the ISP effectively. Our algorithm performs efficiently in a wide range of different cases without the need to tune its predefined parameters. It is shown to outperform the state-of-the-art approaches in simulations. Even though BOIS 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 pay for performance or using bonuses). Although BOIS is an important initial step towards solving the ISP, there are some areas of further work. First, we assume that time steps are homogeneous and a new incentive can be started only when all previous ones have completed. 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 exploring, especially when the time budget is limited. Second, we also asume that the cost of applying an incentive is the same at all times. This may be limiting in more general settings. For example, some incentives are inherently designed with variable payment such as pay for performance or using bonuses. Third, the model of user performance used in the simulations is rather simple, while it might be more complicated in different projects. Thus, running experiments might help us better understand how people behave in different cases. Hence, we can design better algorithms to solve the ISP more efficiently. Regarding other applications of our work, the model 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).

Acknowledgments

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorised to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Bibliography

Araujo, R.M.: 99designs: An analysis of creative competition in crowdsourced design. In: HCOMP. pp. 17–24 (2013)

Badanidiyuru, A., Kleinberg, R., Slivkins, A.: Bandits with knapsacks. JACM 65(3), 1–55 (2018)

Bubeck, S., Stoltz, G., Szepesvári, C., Munos, R.: X-armed bandits. JMLR 12, 1655–1695 (2011)

Cavallo, R., Jain, S.: Efficient crowdsourcing contests. In: AAMAS. vol. 2, pp. 677–686 (2012)

Doan, A., Ramakrishnan, R., Halevy, A.Y.: Crowdsourcing systems on the world-wide web. CACM 54(4), 86–96 (2011)

Frey, B.S., Jegen, R.: Motivation crowding theory. Journal of Economic Surveys 15(5), 589–611 (2001)

Ghezzi, A., Gabelloni, D., Martini, A., Natalicchio, A.: Crowdsourcing: A review and suggestions for future research. IJMR 20(2), 343–363 (2018)

Ho, C.J., Slivkins, A., Vaughan, J.W.: Adaptive contract design for crowdsourcing markets: Bandit algorithms for repeated principal-agent problems. JAIR 55, 317–359 (2016)

Johnson, M., Moore, L., Ylvisaker, D.: Minimax and maximin distance designs. JSPI 26(2), 131–148 (1990)

Li, H., Xia, Y.: Infinitely many-armed bandits with budget constraints. In: AAAI. pp. 2182–2188 (2017)

Luo, T., Kanhere, S.S., Tan, H.P., Wu, F., Wu, H.: Crowdsourcing with tullock contests: A new perspective. In: INFOCOM. pp. 2515–2523 (2015)

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

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

Rogstadius, J., Kostakos, V., Kittur, A., Smus, B., Laredo, J., Vukovic, M.: An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In: ICWSM. pp. 321–328 (2011)

Simula, H.: The rise and fall of crowdsourcing? In: HICSS. pp. 2783–2791 (2013)

Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: NIPS. p. 9 (2012)

Tran-Thanh, L., Chapman, A., De Cote, E.M., Rogers, A., Jennings, N.R.: Epsilon-first policies for budget-limited multi-armed bandits. In: AAAI. pp. 1211–1216 (2010)

Trovo, F., Paladino, S., Restelli, M., Gatti, N.: Budgeted multi-armed bandit in continuous action space. In: ECAI. pp. 560–568 (2016)

Truong, N.V.Q., Stein, S., Tran-Thanh, L., Jennings, N.R.: Adaptive incentive selection for crowdsourcing contests. In: AAMAS. pp. 2100–2102 (2018)

Yin, M., Chen, Y.: Bonus or not? Learn to reward in crowdsourcing. In: IJCAI. pp. 201–207 (2015)