

UNIVERSITY OF SOUTHAMPTON

Incentive Engineering in Microtask Crowdsourcing

by

Nhat Van-Quoc Truong

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

November 2020

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Nhat Van-Quoc Truong

Crowdsourcing is emerging as an efficient approach to solve a wide variety of problems by engaging a large number of Internet users from many places in the world. However, the success of these systems relies critically on motivating the crowd to contribute, especially in microtask crowdsourcing contexts when the tasks are repetitive and easy for people to get bored. Given this, finding ways to efficiently incentivise participants in crowdsourcing projects in general and microtask crowdsourcing projects in particular is a major open challenge. Also, although there are numerous ways to incentivise participants in microtask crowdsourcing projects, the effectiveness of the incentives is likely to be different in different projects based on specific characteristics of those projects. Therefore, in a particular crowdsourcing project, a practical way to address the incentive problem is to choose a certain number of candidate *incentives*, then have a good strategy to select the most effective incentive at run time so as to maximise the cumulative utility of the requesters within a given budget and time limit. We refer to this as the *incentive selection problem* (ISP).

We present algorithms (HAIS and BOIS) to deal with the ISP by considering all characteristics of the problem. Specifically, the algorithms make use of limited *financial and time budgets* to have a good exploration-exploitation balance. Also, they consider the *group-based nature* of the incentives (i.e., sampling two incentives with different group size yields two different number of samples) so as to make a good decision on how many times each incentive will be sampled at each time. By conducting extensive simulations, we show that our algorithms outperform state-of-the-art approaches in most cases. Also from the results of the simulations, practical usage of the two algorithms is discussed.

Contents

| | |
|--|-------------|
| List of Figures | viii |
| List of Tables | x |
| Nomenclature | xii |
| Declaration of Authorship | xvii |
| Acknowledgements | xix |
| 1 Introduction | 1 |
| 1.1 Motivation of the Research | 1 |
| 1.2 Research Objectives | 7 |
| 1.3 Research Contributions | 9 |
| 1.4 Thesis Outline | 11 |
| 2 Literature Review | 13 |
| 2.1 Motivation and Incentives in Microtask Crowdsourcing | 13 |
| 2.1.1 Motivation | 14 |
| 2.1.2 Incentives | 16 |
| 2.1.3 Interactions between Incentives and Motivation | 20 |
| 2.2 Online Learning | 24 |
| 2.2.1 Incentive Selection Problem | 24 |
| 2.2.2 Multi-armed Bandits | 24 |
| 2.2.2.1 Budget Constraints | 25 |
| 2.2.2.2 Batched Pulling | 27 |
| 2.2.2.3 Crowdsourcing | 28 |
| 2.2.2.4 Many-armed Bandits | 29 |
| 2.2.3 Bayesian Optimisation | 31 |
| 2.2.3.1 Gaussian Process Prior | 32 |
| 2.2.3.2 Acquisition Functions | 34 |
| 2.3 Summary | 36 |
| 3 The Incentive Selection Problem (ISP) | 39 |
| 3.1 The Problem | 39 |
| 3.2 Three Variants of the ISP | 41 |
| 3.2.1 The ISP1 (Uncorrelated ISP) | 41 |
| 3.2.2 The ISP2 (Correlated ISP) | 42 |

| | | |
|----------|--|------------|
| 3.2.3 | The ISP3 (Mixed-correlated ISP) | 42 |
| 3.3 | Measuring the Utility of the Requesters | 43 |
| 3.4 | Measuring the Effectiveness of the Incentives | 44 |
| 4 | The ISP with Uncorrelated Candidate Incentives (ISP1) | 47 |
| 4.1 | The Problem | 47 |
| 4.2 | The HAIS Algorithm | 49 |
| 4.2.1 | The Hoeffding's Inequality | 49 |
| 4.2.2 | Algorithm Overview | 50 |
| 4.2.3 | The Sampling Step | 53 |
| 4.2.4 | The Hoeffding Step | 54 |
| 4.2.5 | The Stepped Exploitation Step | 59 |
| 4.2.6 | The Pure Exploitation Step | 59 |
| 4.3 | Experimental Evaluation | 60 |
| 4.3.1 | Benchmarks | 60 |
| 4.3.2 | Simulation Settings | 63 |
| 4.3.2.1 | Ranges of the Quantities for Randomisation | 64 |
| 4.3.2.2 | Values of the Predefined Parameters of the Algorithms | 67 |
| 4.3.2.3 | The Model of Group Performance | 68 |
| 4.3.3 | Results | 68 |
| 4.3.3.1 | Exploration-exploitation Balance | 73 |
| 4.3.3.2 | Taking Advantage of the Time Budget | 73 |
| 4.3.3.3 | Effective Elimination | 75 |
| 4.3.4 | Practical Usage of the HAIS Algorithm | 77 |
| 4.4 | Summary | 77 |
| 5 | The ISP with Correlated Candidate Incentives (ISP2) | 83 |
| 5.1 | The Problem | 84 |
| 5.2 | The BOIS Algorithm | 84 |
| 5.2.1 | Algorithm Overview | 85 |
| 5.2.2 | The Sampling Step | 88 |
| 5.2.3 | The Stepped Exploitation Step | 88 |
| 5.2.4 | The Pure Exploitation Step | 90 |
| 5.3 | Experimental Evaluation | 90 |
| 5.3.1 | Benchmarks | 91 |
| 5.3.2 | Simulation Settings | 92 |
| 5.3.2.1 | Ranges of the Quantities for Randomisation | 92 |
| 5.3.2.2 | Values of the Predefined Parameters of the Algorithms | 93 |
| 5.3.3 | Results | 95 |
| 5.3.3.1 | Taking Advantage of the Time Budget | 95 |
| 5.3.3.2 | Effective Sampling | 98 |
| 5.3.4 | Practical Usage of the BOIS Algorithm | 98 |
| 5.3.5 | When to use HAIS and BOIS Algorithms to deal with the ISP | 99 |
| 5.4 | Summary | 100 |
| 6 | Conclusions and Future Work | 103 |
| 6.1 | Summary | 103 |

| | |
|-----------------------------------|------------|
| 6.2 Future Work | 106 |
| A User Motivation Theories | 109 |
| Bibliography | 115 |

List of Figures

| | | |
|------|---|-----|
| 1.1 | Illustrative Examples of Correlations between the Incentives in a Cluster | 5 |
| 2.1 | Net-outcome of the Price- and the Crowding-Out-Effect | 22 |
| 2.2 | An Illustrative Example of How Bayesian Optimisation Works | 33 |
| 4.1 | An Example of an Applying Policy | 48 |
| 4.2 | An Example of Running HAIS | 52 |
| 4.3 | Illustration for Equation 4.10. | 57 |
| 4.4 | An Example of Applying Policy with Stepped ϵ-first | 60 |
| 4.5 | Performance of Algorithms for Different Values of B | 69 |
| 4.6 | Performance of Algorithms for Different Values of T | 69 |
| 4.7 | Performance of Algorithms for Different Values of I | 70 |
| 4.8 | Performance of Algorithms for Different Values of σ_i | 70 |
| 4.9 | Performance of Algorithms for Different Values of Max Group Size | 71 |
| 4.10 | Performance of Algorithms for Different Values of Max Group Size on the Best Incentive | 71 |
| 4.11 | Performance of Algorithms for Different Values of Max Group Size on the Worst Incentive | 72 |
| 4.12 | Average Budget Used for Exploration | 74 |
| 4.13 | Cost Distribution Over the Incentives Across the Phases of Each Algorithm | 75 |
| 4.14 | Cost Distribution Across the Periods Incurred by Each Algorithm | 78 |
| 4.15 | The first three periods of Figure 4.14a | 79 |
| 4.16 | Performance of HAIS for Different Budget Ratios | 80 |
| 4.17 | Number of Periods Used by HAIS | 80 |
| 5.1 | An Illustration of Candidate Incentives in a Cluster in the Sampling Step | 87 |
| 5.2 | Performance of Algorithms for Different Values of T | 96 |
| 5.3 | Performance of Algorithms for Different Values of B | 96 |
| 5.4 | Performance of Algorithms for Different Values of C | 97 |
| 5.5 | Performance of Algorithms for Different Values of σ_a | 97 |
| 5.6 | Performance of BOIS for Different Budget Ratios | 99 |
| A.1 | User Motivation Theories used in building the Incentive Map | 110 |

List of Tables

| | | |
|-----|---|----|
| 4.1 | Ranges for Randomisation of the Parameters in the Simulations | 65 |
| 4.2 | Values of the Algorithms' Predefined Parameters in the Simulations. | 67 |
| 5.1 | Ranges for Randomisation of the Parameters in the Simulations | 93 |
| 5.2 | Values of the Algorithms' Predefined Parameters in the Simulations. | 94 |

Nomenclature

General

| | | |
|----------------------------|--|--|
| \sim | Distributed according to | |
| \approx | Approximately equal to | |
| $\stackrel{\text{def}}{=}$ | Is equal by definition to | |
| \mathbb{E} | Expectation | |
| \mathbb{R} | The real numbers | |
| \mathcal{N} | Normal (Gaussian) distribution | |
| Φ | Cumulative density function of standard normal distribution | |
| ϕ | Probability density function of standard normal distribution | |
| API | Application programming interface | |
| BO | Bayesian optimisation | |
| EI | Expected improvement | |
| GP | Gaussian process | |
| MAB | Multi-armed bandit | |
| PI | Probability of improvement | |
| TS | Thompson sampling | |
| UCB | Upper confidence bound | |
| ISP | ISP | Incentive selection problem |
| | ISP1 | (or Uncorrelated ISP) ISP with uncorrelated candidate incentives |
| | ISP2 | (or Correlated ISP) ISP with correlated candidate incentives |
| | ISP3 | (or Mixed-correlated ISP) ISP with mixed-correlated candidate incentives |
| | B | Financial budget |
| T | Time budget | |
| Algorithms | BOIS | Bayesian-optimisation based incentive selection |
| | H AIS | Hoeffding based adaptive incentive selection |
| | KUBE | Knapsack-based UCB exploration and exploitation |
| | fKUBE | Fractional KUBE |
| | sfKUBE | Stepped fKUBE |
| | ϵ -greedy | Decaying ϵ -greedy |
| | sc-first | Stepped ϵ -first |
| SOAAv | Survival of the above average | |

Chapter 2

| | | | |
|------------|---|-----------------------------------|---|
| { | Incentive Map | A | Action |
| | | O | Design objective |
| | | P | Population aspect |
| | | AO | Action-design objective connection |
| | | OP | Design objective-population aspect connection |
| | | AO table | Connection table between actions and design objectives |
| | | OP table | Connection table between design objectives and population aspects |
| ϵ | Budget for exploration in ϵ -first and Stepped ϵ -first | | |
| { | BO | f | Unknown costly objective function |
| | | \mathcal{X} | Input space |
| | | \mathbf{x} | Input: $\mathbf{x} \in \mathcal{X}$ |
| | | \mathbf{x}_i | The i th input: $\mathbf{x}_i \in \mathcal{X}$ |
| | | y | Output |
| | | y_i | The i th output |
| | | y^+ | The best output at time step t : $\mathbf{y}^+ = \max_{\mathbf{x}_{1:n_t}} f(\mathbf{x}_i)$ |
| | | \mathbf{y} | Output vector whose i th value is y_i |
| | | ϵ_{noise} | Output noise: $y = f(\mathbf{x}) + \epsilon_{\text{noise}}$ |
| | | σ_{noise}^2 | Noise variance |
| | | T | Time horizon |
| | | n_0 | Number of samples specified by a space-filling design |
| | | n_t | Number of samples at time step t : $n_t = n_0 + t$ |
| | | \mathcal{D}_t | Data set at time step t : $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_t}$ |
| | | \mathcal{M} | Surrogate model |
| | | \mathcal{M}_t | Surrogate model updated at time step t |
| | | \mathcal{GP} | GP surrogate model |
| | | \mathcal{GP}_t | GP surrogate model updated at time step t |
| | | μ_0 | Mean function |
| | | μ_t | Posterior function at time step t |
| | | σ_t^2 | Variance function at time step t |
| | | κ | Covariance function (or kernel) |
| | | $\kappa(\mathbf{x}, \mathbf{x}')$ | Covariance function evaluated at \mathbf{x} and \mathbf{x}' |
| | | \mathbf{K} | Covariance matrix: $\mathbf{K}_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ |
| | | \mathbf{I} | The identity matrix |
| | | $\mathbf{k}_t(\mathbf{x})$ | $(n_t \times 1)$ vector whose i th value is $\kappa(\mathbf{x}, \mathbf{x}_i)$ |
| | | α | Acquisition function |
| | | α_{EI} | EI acquisition function |
| | | α_{PI} | PI acquisition function |
| | | α_{TS} | TS acquisition function |

| | | |
|---|-----------------------|--|
| { | α_{UCB} | UCB acquisition function |
| | λ | Tunable parameter used in UCB acquisition function |
| | τ | A target to be used in EI and PI acquisition functions |

Chapter 4

| | | |
|--------------|---------------------|---|
| { | I | Number of incentives |
| | \mathcal{I} | Set of incentives |
| | μ_i | The expected utility of incentive i |
| | σ_i | The standard deviation of incentive i 's utility |
| | $\hat{\mu}_i^{(t)}$ | The estimate of μ_i at the end of period t |
| | c_i | The cost of incentive i |
| | δ_i | The density (i.e., effectiveness) of incentive i |
| | $d_i^{(t)}$ | The estimate of incentive i 's density at the end of period t |
| | $n_i^{(t)}$ | The number of times incentive i is applied in period t |
| | $r_i^{(t)}$ | The utility of applying incentive i in period t |
| | $u_i^{(t)}$ | The number of sampled users in incentive i until the end of period t |
| | \mathbf{N} | An applying policy: $\mathbf{N} = \{n_i^{(t)} \mid t = 1, \dots, T; i = 1, \dots, I\}$ |
| | ϵ_1 | Budget for exploration in ϵ-first and Stepped ϵ-first |
| | { | ϵ_1 |
| ϵ_2 | | Budget limit for stepped exploitation in HAIS and also in Stepped ϵ-first and Stepped fKUBE |
| U_1 | | Target number of sampled users to obtain in each incentive after period 1 |
| U_2 | | Target number of sampled users to obtain in each active incentive after period 2 |
| L_e | | Confidence level to eliminate ineffective incentives |
| L_h | | Confidence level to stop exploring |
| L_s | | Confidence level to stop stepped exploiting |

Chapter 5

| | | |
|---|---------------------|---|
| { | C | Number of clusters |
| | C_i | Cluster i |
| | K_i | Number of parameters of C_i |
| | μ_a | The expected utility of incentive a |
| | σ_a | The standard deviation of incentive a 's utility |
| | $\hat{\mu}_a^{(t)}$ | The estimate of μ_a at the end of period t |
| | c_a | The cost of incentive a |
| | δ_a | The density (i.e., effectiveness) of incentive a |
| | $d_a^{(t)}$ | The estimate of incentive a 's density at the end of period t |

| | | | |
|----------------------|---|--------------|---|
| ISP2 | { | $d_a^{*(t)}$ | The potential effectiveness of incentive a at the end of period t |
| | | $v_a^{(k)}$ | The value of the k th parameter of incentive a |
| | | v_a | Structure vector corresponding to incentive a : $v_a = (v_a^{(1)}, \dots, v_a^{(K_i)})$ |
| | | $n_a^{(t)}$ | The number of times incentive a is applied in period t |
| | | $r_a^{(t)}$ | The utility of applying incentive a in period t |
| | | \mathbf{N} | An applying policy: $\mathbf{N} = \{n_a^{(t)} \mid t = 1, \dots, T; a \in C_i; i = 1, \dots, C\}$ |
| s- ϵ -first | { | ϵ_1 | Budget for exploration |
| | | ϵ_2 | Budget for stepped exploitation |
| | | U_1 | Target number of sampled users to obtain in each incentive after period 1 |
| BOIS | { | ϵ_1 | Budget limit for exploration |
| | | ϵ_2 | Budget limit for stepped exploitation in BOIS and also in Stepped ϵ -first |
| | | U_1 | Target number of sampled users to obtain in each incentive after period 1 |
| | | U_2 | Target number of sampled users to obtain in each active incentive after period 2 |
| | | D_{min} | Confidence level to eliminate ineffective incentives |
| | | Z | Critical value corresponding to the confidence level in eliminating ineffective incentives |

Declaration of Authorship

I, Nhat Van-Quoc Truong, declare that the thesis titled “Incentive Engineering in Microtask Crowdsourcing”, and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- This work was done wholly while in candidature for a research degree at the University;
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any institution, this has been clearly stated;
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- Parts of this work have been published in a number of conference and journal papers (see Section 1.3 for a list).

Signed:

Date:

Acknowledgements

Undertaking this PhD has been a truly life-changing experience for me and it would not be possible to do it alone.

First and foremost, I would like to express my deepest gratitude to my supervisors, Sebastian Stein, Long Tran-Thanh, and Nick Jennings, for their guidance, support, encouragement, and patience throughout my PhD. I am incredibly thankful for their investment of a tremendous amount of time not only during weekly meetings but also while correcting my writings even over weekends and holidays.

I would also like to thank my lab mates, my friends in Southampton and Hue cities for their friendship, “distractions”, discussions, and support. In this respect, I owe special thanks to Quang Le, Hang Phan, Henry Cuong, Dong Huynh, Phuoc Doan, Phuong Ngo, Nhan Luong, Tra Hoang, Md. Mosaddek Khan (Sajeeb), Olabambo Oluwasuji, Belfrit Batlajery, and Alexandry Augustin.

I gratefully acknowledge the financial support provided by the 911 Project of the Vietnamese Government, the Faculty of Engineering and Physical Sciences – University of Southampton, and the DAIS ITA project. I am also thankful to the IRIDIS High Performance Computing Services for helping me complete my simulations.

I would also like to thank Hue University of Sciences and especially Department of Information Technology for their assistance and encouragement during my doctoral studies.

Last, but no means the least, I thank my entire family for their endless love and unconditional support. My special thanks go to my wife, Huong, who has stood by me throughout the phases of my PhD and my son, Nguyen, for giving me new energy every day.

*To my family
for their love, support and encouragement.*

Chapter 1

Introduction

With the development and popularity of the Internet, web 2.0 applications and smart devices, people from all corners of the world can be easily gathered together in a virtual space to conduct activities, such as helping build the maps of a disaster area for humanitarian aid or helping scientists in classifying galaxies according to their shapes. In fact, a virtual crowd of non-specialised people (individuals who are not specialised in the problems of interest) assembled in this way can sometimes accomplish tasks more quickly and at a lower cost (with equivalent or better quality) than computers or experts (Callison-Burch, 2009; Snow et al., 2008; Zook et al., 2010). However, motivating them to participate in such *crowdsourced* projects (projects that are conducted by a crowd via the Internet) is still a big challenge, especially when tasks are not intrinsically interesting, as is often the case with microtask crowdsourcing projects. *Microtask* crowdsourcing projects are crowdsourcing projects whose tasks are small and can be completed in seconds or minutes, such as translating a few sentences, transcribing a short audio clip, or identifying objects in an image. In this thesis, we focus on this challenge of finding ways to encourage contributors in a microtask crowdsourcing project to perform tasks. We start the chapter by describing the motivation and challenges of the research (Section 1.1). Then, we state the research objectives (Section 1.2). After that, we present the contributions of our research to date (Section 1.3). Finally, we introduce the structure of the thesis (Section 1.4).

1.1 Motivation of the Research

Attracting a multitude of disparate individuals to join a project or a community is one of the most intriguing and promising features of the Internet (Mason and Watts, 2010). *Crowdsourcing*¹ takes advantage of this feature by soliciting the crowd through an open call in dealing with a wide variety of problems (Doan et al., 2011; Ghezzi et al., 2018).

¹The term was coined by Jeff Howe in a June 2006 article of *Wired* magazine (Howe, 2006).

For example, crowd members can contribute new ideas to a product, provide solutions for a problem, or vote for an opinion (Simula, 2013). Crowdsourcing attracts organisations and companies not only because it provides cheap labour, but also because it helps to solve problems quickly with high quality (Callison-Burch, 2009; Snow et al., 2008; Zook et al., 2010). Thus, recently there has been significant interest in the research community in building autonomous agents to run crowdsourcing projects efficiently (Biswas et al., 2015; Cavallo and Jain, 2012; Itoh and Matsubara, 2016; Jain et al., 2016; Kamar and Horvitz, 2012; Kara et al., 2018; Sen et al., 2015; Tran-Thanh et al., 2014; Venanzi et al., 2016). However, the success of crowdsourcing projects relies critically on motivating a crowd² to contribute (Doan et al., 2011; Simula, 2013). In microtask crowdsourcing projects, this is more challenging when the tasks are not intrinsically interesting. The tasks in these projects are simple and usually do not require any knowledge or techniques to complete. Hence, after performing these tasks repeatedly, users are likely to get bored.

There are extensive studies of motivation and incentives from many fields of study, such as psychology, sociology, organisational behaviour, marketing, and computer science under various perspectives, such as humans in general, workers in firms, customers of brands, members in social groups or online communities, and contributors in crowdsourcing systems. From these studies, various *incentive methods* can be considered for use to align the behaviours of users with the objectives of requesters³ in crowdsourcing projects. For example, we can use methods related to the connections and communications between the users of the crowdsourcing project, such as providing environments for the users to communicate (Kraut et al., 2011) or allowing them to give feedback on the tasks (Kulkarni et al., 2015). We can also use methods related to the tasks to arouse the intrinsic motivations of the users. An example is showing pop-up messages before users are predicted to disengage times of a user in a work session to prolong the session and hence to achieve higher task completion rates (Avi et al., 2016). Another example is providing a wide variety of tasks to prevent users from becoming bored while performing a certain type of tasks for a long type (Alsayasneh et al., 2018; Kaufmann et al., 2011). A different group of incentive methods, which is widely used in microtask crowdsourcing, is rewarding users for completing tasks. For instance, users will be rewarded a certain amount of money (e.g., £2.00) for completing a certain number of tasks (e.g., 20). This incentive method is called paying for performance (Mason and Watts, 2010; Prendergast, 1999). We can also sometimes give them a bonus (e.g., £1.00) for

²In the context of crowdsourcing, the *crowd* is sometimes referred to using several different names. Under the view of a component of a crowdsourcing system, they can be called *crowdsourcers*, while with the view of a company, we can call them *workers*. However, when considering them as the ones who contribute to an organisation or a project, we can use the word *contributors* or *participants*. And finally, they are members of the community and also users of the system, thus we can call them *members* or *users*. As a consequence, when considering a specific case (e.g., a specific system), we use appropriate names. For example, when talking about a citizen science project, we use the word *contributors* or *participants* (instead of *workers*). And, in general cases, we use the word “users”.

³In some projects, the requesters can also be the designers. To keep the presentation simple, we use “requesters” to refer to both.

completing the tasks with a high quality (Harris, 2011; Yin and Chen, 2015). These and many other incentive methods are discussed in Chapter 2.

Another effective method to incentivise users in crowdsourcing is using *contests*⁴ as they are effective and cheap. Actually, by rewarding users in a contest, task requesters do not necessarily have to pay for every task completed as in other types of financial rewards, such as paying for performance or using bonuses. Indeed, they have to pay only for a certain number of users, e.g., the top user who has completed the tasks with the highest quality or the top two who have completed the most tasks. 99designs (www.99designs.com), TopCoder (www.topcoder.com), and Taskcn (www.taskcn.com) are some well-known crowdsourcing platforms that use contests to attract users.

Although there are numerous incentives that can be used in a specific crowdsourcing project, finding ways to efficiently incentivise users is still an open challenge. This is because the effectiveness of the incentives is usually unknown in advance and is likely to be different in different projects based on specific characteristics of those projects. For example, they can be 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 user community (e.g., the extent to which they are in contact with each other or the extent to which the information of a user is exposed to the others in the community). These differences reflect the fact that users have different motivations (Frey and Jegen, 2001; Heyman and Ariely, 2004; Zheng et al., 2011). Also, the structure (i.e., the group size and the rewards) of the contests can make a significant difference in performance (Feyisetan and Simperl, 2019; Ramchurn et al., 2013). Thus, more work is needed to establish efficient and practical approaches that crowdsourcing designers, the ones who actually implement and deploy the projects, can use to incentivise users to perform tasks so as to maximise the overall utility of the requesters. We refer to this as the *incentive problem*. In a microtask crowdsourcing project, the *utility* of the requester on a task is the benefit that the requester receives after the task is completed which can be measured by including task quantity, task quality, task completion time, or some subset of them. The detailed discussion about utility measurement will be presented in Section 3.3.

One practical way to solve the incentive problem is, first, choosing a certain number of candidate *incentive methods* suggested by related studies and from prior knowledge about the crowdsourcing projects and then, having a good approach to select the most effective incentive method and identify the most effective incentive in the method. To clarify the concept of incentive used in our research, an incentive method (such as paying for performance or contests) might correspond to several incentives as they might have one or more parameters and each parameter may have a range of possible values. An example is that, paying for performance (an incentive method) might have one

⁴We use the term “contest” in a broad sense to refer to any situation in which users 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 work.

parameter, the amount of money paid for completing, say, 50 tasks. The payment can range from £5.00 to £10.00. Another example is that contests might have two parameters, the group size (the maximum number of users in a contest, a parameter) and the amount of prize money for the best user. The group size might range from, say, 5 to 30, while the prize for the best user can be from £2.00 to £10.00. A candidate incentive corresponds to an incentive method with specific values of the parameters. For ease of presentation, we refer to an incentive method as a cluster, as it is a cluster of incentives. Specifically, a *cluster* is a group of incentives corresponding to an incentive method but having different values of the parameters. The chosen incentives in each cluster are referred to as *candidate incentives*. When it does not lead to confusion, we simply use “incentives” instead of “candidate incentives”.

Within a cluster, there might be correlations between the incentives. This means the difference in the effectiveness between two adjacent incentives (i.e., the values of their parameters are slightly different) is small. Indeed, it is likely that when the parameter values of the incentive method change slightly, the effectiveness of the corresponding incentives also changes gradually. In other words, the *correlation* between two incentives in a cluster is the extent to which difference in effectiveness of the incentives based on the values of their parameters. Figure 1.1 shows possible correlations between the incentives in two clusters corresponding to the two incentive methods mentioned above. Specifically, Figure 1.1a shows the effectiveness of the incentives, measured by utility per cost unit⁵, in cluster 1 (corresponding to the incentive method paying for performance). This figure depicts that utility initially increases with increasing payment. However, when it is larger than £8.00, the effectiveness starts decreasing. Similarly, Figure 1.1b shows possible correlations in the second cluster. However, if we continue choosing some incentives in each cluster, the chosen incentives are not correlated. For example, in cluster 1, if we continue choosing only two incentives, a_1 and a_2 (other incentives in the cluster are not candidate incentives any more), these two incentives are not correlated. Similarly, in cluster 2, if we continue choosing only three incentives, b_1 , b_2 , and b_3 , these three incentives are not correlated. Later in the section, we will discuss the reason, in some cases, we have to continue choosing some incentives in each cluster.

Furthermore, currently on many microtask crowdsourcing platforms, such as Amazon Mechanical Turk (www.mturk.com), Clickworker (www.clickworker.com), and Figure Eight (www.figure-eight.com, formerly known as CrowdFlower), 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 users or sending bonuses to users with high quality submissions) in an autonomous manner using programmable Application Programming Interfaces (APIs). This makes it possible to build autonomous agents to monitor and adaptively switch incentives when appropriate.

⁵The measurement of an incentive’s effectiveness will be discussed in more detail in Section 3.4.

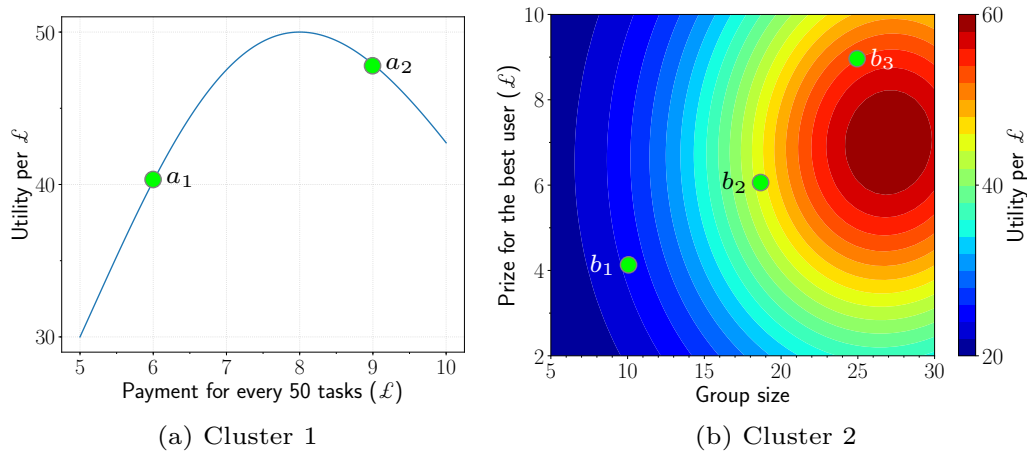


FIGURE 1.1: Illustrative examples of correlations between the incentives in a cluster which has one (a) and two (b) parameters. The green points are the chosen incentives when we have prior knowledge about the project or the budget for the project is small.

Indeed, it is inconvenient, or almost impossible in many cases, to switch between incentives manually to identify the best one, especially in microtask crowdsourcing, where the number of tasks and the number of users are usually large. Therefore, it is a key problem to find an appropriate way for an autonomous agent to select effective incentives in a microtask crowdsourcing project so as to maximise the requester’s overall utility within a given budget. We refer to this as the *incentive selection problem* (ISP).

As mentioned above, the effectiveness of the incentives in a specific crowdsourcing project is usually 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) form a promising approach for this problem. Specifically, this approach models the problem as a machine with I arms (corresponding to I incentives), pulling an arm (providing the corresponding incentive to a group of users) 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 a budgeted MAB problem (corresponding to the ISP) is to find a pulling policy 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 (Badanidiyuru et al., 2018; Sen et al., 2015; Tran-Thanh et al., 2012, 2010; Zhou and Tomlin, 2018). However, these algorithms are not designed to work with the *time budget* (i.e., the deadline) of the ISP. Moreover, the studies do not consider the *group-based nature* of the incentives when they are contests. In detail, after pulling an arm, we receive the utilities of the individuals in the corresponding contest group (i.e., a number of data points) rather than the total utility of the whole group (i.e., one data point). Thus, as we will show in Chapter 4, 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 users 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., users). As a result, the second case requires fewer pulls of exploration in order to achieve the same level of understanding of the users' performance. For example, after 5 pulls of an arm in the second case, we have 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, in order to determine the appropriate numbers of pulls for the arms. Additionally, another important characteristic of the ISP is *batched pulling*. That is, in each round, multiple arms can be pulled (i.e., several incentives can be offered at a time) and an arm can be pulled multiple times (i.e., an incentive can be offered to different user groups). Many other studies have investigated batched MABs (Anantharam et al., 1987; Audibert et al., 2014; Gao et al., 2019; Jun et al., 2016; Luedtke et al., 2019; Perchet et al., 2016; Zhou and Tomlin, 2018; Xia et al., 2016). Yet, none of these studies examine both the batched and budgeted characteristics of the ISP. For example, in the models of Audibert et al. (2014); Gao et al. (2019); Jun et al. (2016); Perchet et al. (2016) and Anantharam et al. (1987), there must be exactly K out of I arms to be pulled in each round, whereas in the ISP, the number of incentives to be offered at a round is arbitrary. Although Luedtke et al. (2019); Zhou and Tomlin (2018) and Xia et al. (2016) consider batched and budgeted MABs, in their models an arm is allowed to be pulled at most once each time. Therefore, this is a non-trivial extension of the MABs and requires new approaches that can deal with all the characteristics of the ISP effectively.

The MAB algorithms previously mentioned deal with the ISP with a small number of candidate incentives, where the budget is large enough to apply all the incentives several times to estimate their effectiveness. This is the case where we have prior knowledge about the crowdsourcing project, thus we can choose some (possibly good) incentive methods, and with each method, we can choose some (possibly good) candidate incentives. That means we can (or have to) determine specific values of the parameters of each chosen incentive method. Figure 1.1 shows examples of some chosen candidate incentives. With the first incentive method (paying for performance), we have chosen two candidate incentives a_1 and a_2 , which correspond to the payment of £6.00 and £9.00 respectively (Figure 1.1a). And with the second incentive method (contests), the chosen incentives are b_1 , b_2 , and b_3 (Figure 1.1b). This is also the case where the budget for the project is small, and hence we have to limit the number of incentives so that we do not have to spend a large amount of money on exploring. However, when we do not have enough prior knowledge about the project and the budget for the project is large enough, we can consider selecting the best incentive from the chosen incentive methods so that we do not miss the best incentive in the best method. The above-mentioned MAB algorithms are not a good way to deal with this case of the ISP, i.e., the case where the candidate incentives are many and correlated. This is because these algorithms need to explore all the arms (i.e., apply all the candidate incentives) at least once in order to

identify the best arm (i.e., the most effective incentive). This is not effective as it may spend a large portion of the budget for exploring the arms, so the remaining budget is too small for exploiting the best arm explored. In some cases, this is impossible as they do not have enough budget for pulling each arm once.

There are many studies about MABs where there are many arms (henceforth, many-armed bandits) (Bubeck et al., 2011; Chaudhuri and Kalyanakrishnan, 2018; Grill et al., 2015; Li and Xia, 2017; Trovo et al., 2016; Wang et al., 2008). Yet, none of them can be used to solve the ISP. Actually, they do not consider all the characteristics of the ISP2, such as the budget constraints, multidimensional structure of the incentives (i.e., an incentive method has a certain number of parameters), correlations between the arms, or the group-based nature of the arms. For these reasons, we are motivated to investigate approaches to deal with the possible variants of the ISP in microtask crowdsourcing. The first variant corresponds to crowdsourcing projects where the incentives in each cluster are uncorrelated. We refer to this variant as the *ISP1* (uncorrelated ISP or the ISP with uncorrelated incentives) and the corresponding projects as *uncorrelated projects*. The second variant corresponds to crowdsourcing projects where the incentives in each cluster are correlated. We refer to this variant as the *ISP2* (correlated ISP or the ISP with correlated incentives) and the corresponding projects as *correlated projects*. The third variant corresponds to crowdsourcing projects where the incentives in some clusters are uncorrelated and the incentives in the other clusters are correlated. We refer to this variant as the *ISP3* (mixed-correlated ISP or the ISP with mixed-correlated incentives) and the corresponding projects as *mixed-correlated projects*. In summary, our research question is:

“How to find appropriate approaches for an autonomous agent to maximise the requesters’ cumulative utility in all (i.e., uncorrelated, correlated, and mixed-correlated) microtask crowdsourcing projects by effectively selecting the right incentives within a given budget and a time limit?”

In the next sections, we will show how we address this question in our research.

1.2 Research Objectives

This research has the following objectives:

RO1: The approaches must be *efficient*. Specifically, they must help requesters of microtask crowdsourcing projects maximise the overall utility by having a good balance between exploration (i.e., learning the effectiveness of the incentives) and exploitation (i.e., applying the most effective incentive). In doing so, they must

make use of the given budget and the time limit to effectively apply appropriate incentives at each time. Also, as contests might be a good incentive method to the projects, the approaches must take advantage of the group-based nature of the incentives to achieve good performance.

RO2: The approaches must be *autonomous*. In more detail, the approaches can be used by autonomous agents to automatically and effectively select the right incentives. In this way, the requesters can deploy their projects on crowdsourcing platform by using provided APIs.

RO3: The approaches must be *adaptive*. More specifically, autonomous agents are responsible for selecting the right incentives. Also, in a different crowdsourcing project, the incentives, user performance corresponding to the provided incentives, the budget, and the deadline might be different; while the requester usually does not have enough prior knowledge about the user performance in the project to provide the agents. Hence, the approaches must be able to adaptively change their behaviours so that the agents can run efficiently in different projects.

RO4: The approaches must be *complete*. That means they are able to deal with the ISP in any crowdsourcing project. In other words, the approaches to the ISP must cover at least the two variants of the problem where the candidate incentives are uncorrelated and correlated.

In order to achieve these objectives, online learning algorithms are a good approach. In detail, budgeted multi-armed bandit (MAB) algorithms can deal with the efficiency (**RO1**) as they are proposed to deal with the exploration-exploitation tradeoff effectively. The Bayesian optimisation (BO) approach might also be a good way to help obtain **RO1** when the candidate incentives are numerous and correlated. Specifically, 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 a given crowdsourcing project, BO can make use of this to find the global optimum more quickly. Also, the MAB algorithms need to be designed to cover group-based (rather than only individual-based) incentives to improve their performance. Therefore, by building algorithms based on MABs and Bayesian optimisation, we can achieve **RO2**, as the agents can apply these algorithms in an automatic manner. To attain **RO3**, the algorithms must be designed to run effectively in different projects without manually tuning their situation-specific parameters. Finally, to accomplish **RO4**, we have two options. One is to build algorithms which perform well on any crowdsourcing projects. That means the algorithms can run effectively on uncorrelated, correlated, and mixed-correlated projects. The other is to build separate algorithms to tackle the variants separately. We chose the second option, as the problems corresponding to the variants are inherently different. Indeed, learning the best incentive in crowdsourcing projects

where there are correlations between incentives is more complicated than in projects where no incentives are correlated. This is because we have to solve not only the learning (i.e., identifying the best incentive method) but also the tuning (i.e., choosing the best parameter values of a method) problems.

1.3 Research Contributions

In solving the ISP, we introduce algorithms which take into consideration the budget limit, the deadline, and the group-based nature of the problem. The ultimate purpose of this work is to build algorithms so that autonomous agents 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, we make the following contributions:

- (1) We formalise the ISP when there are several candidate incentives as a novel *batched 2d-budgeted group-based MAB* problem. We refer to this as the ISP1. Here, “batched” means in each round, multiple arms can be pulled and an arm can be pulled multiple times. And “d” means “dimension”. The two dimensions are the financial budget and the time budget (i.e., the deadline). Finally, “group-based” means the group-based nature of the arms.
- (2) We introduce **H AIS**, a novel adaptive algorithm to solve the ISP1 effectively by utilising all the characteristics of the problem. Specifically, it makes use of the limited (financial and time) budgets to have a good balance between exploration and exploitation. Also, the algorithm has an efficient mechanism for eliminating clearly ineffective arms (i.e., incentives) so that it can spend more budget on exploring (together with exploiting) effective ones. Furthermore, it takes account of the batched pulling and especially the group-based nature of the arms to decide how many times an arm is pulled each time. Moreover, **H AIS** can be applied to any crowdsourcing projects without tuning any situation-specific parameters. This is because it can identify an appropriate number of pulls on the arms each time based on the estimates of the arms so far.
- (3) We empirically demonstrate that **H AIS** is generally more effective and efficient compared to state-of-the-art approaches in an extensive series of simulations. The results of the simulations show that **H AIS** outperforms state-of-the-art benchmarks in most cases. Specifically, the performance of **H AIS** is up to 99% of the optimal solution and up to 35% better than state-of-the-art benchmarks. This is due to the following reasons. First, **H AIS** has an efficient strategy in balancing between exploration and exploitation. Second, **H AIS** takes advantage of the time budget to conduct more exploration while exploiting the best incentive so far. Third, the algorithm has an effective elimination mechanism so that it can eliminate clearly

ineffective incentives and hence it can spend more budget on exploring effective ones.

- (4) We formalise the ISP when there are many candidate incentives which are correlated and refer to this as the ISP2. In more detail, we consider an incentive method as a cluster (of the incentives) with parameters whose values are in specific ranges. An incentive (i.e., an arm) is a combination of specific values of the parameters in a cluster. It is assumed that there are correlations between incentives in a cluster. Similar to the ISP1, the ISP2 has the same set of characteristics which are batched pulling, 2d-budgeted, and group-based nature of the incentives.
- (5) We introduce **BOIS**, a novel algorithm to solve the ISP2 effectively. More specifically, different from the ISP1, the ISP2 has two sub-problems. The first one is learning to identify the best cluster and the second one is tuning which is the best incentive in the best cluster. Hence, under limited budgets, we combine MABs and BO in a single process to deal with the two sub-problems effectively. Concretely, we use a MAB approach to learn the best cluster and BO with Gaussian processes to tune the parameters of each cluster.
- (6) We empirically demonstrate that **BOIS** is generally more effective and efficient compared to state-of-the-art approaches in a series of simulations. Specifically, the performance of **BOIS** is up to 92% of the optimal solution and up to 48% better than state-of-the-art benchmarks. This is mainly because **BOIS** makes use of the periods before the deadline to gradually determine the best incentive by combining a MAB technique and BO with Gaussian processes.

Parts of this work have been published in the following venues:

- “Incentive Engineering Framework for Crowdsourcing Systems” at the 4th AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2016 ([Truong et al., 2016](#)). This workshop paper briefly introduced the incentive engineering framework, which tries to summarise and present as a literature map the incentives and motivation factors of users in crowdsourcing projects collected from existing studies.
- “Adaptive Incentive Selection for Crowdsourcing Contests” at the 17th International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2018 ([Truong et al., 2018](#)). This is a short paper that introduced the novel idea of homogeneous-group-size version of the ISP1 (i.e., all incentives have the same group size). The paper introduced the ISP1 and an algorithm to solve this version of the ISP. We also briefly evaluated the performance of the algorithm in the paper. Concretely, the model, i.e., the ISP1, presented in this paper is a simpler version of the ISP1 mentioned in Contribution 1, where the group sizes of

the candidate incentives are homogeneous. Also, the algorithm proposed, **H AIS**, is a simpler version of the **H AIS** algorithm in this thesis which is mentioned in Contribution 2. Hence, the simulations conducted only cover the cases where the group sizes are homogeneous whereas the ones conducted in this thesis also cover heterogeneous-group-size cases (Contribution 3). This paper is the preliminary of the model presented in Chapter 4.

- “What Prize Is Right? How to Learn the Optimal Structure for Crowdsourcing Contests” at the 16th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2019 (Truong et al., 2019). This publication introduced the ISP2, the ISP where the candidate incentives are many (compared to the financial budget) and there exist correlations between the candidates. This is the one mentioned in Contribution 4. We proposed a novel algorithm, **BOIS**, to effectively identify the best cluster and the best incentive in this cluster (Contribution 5). We also evaluated the performance of **BOIS** in the paper (Contribution 6). This paper is primarily the content of Chapter 5.

In addition, parts of this work have been submitted to :

- The Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS). The content of this submission is basically that of Chapter 4. In more detail, it presented the ISP1 (Contribution 1), the **H AIS** algorithm (Contribution 2), and extensive simulations to evaluate the performance of **H AIS** (Contribution 3). This submission is primarily the content of Chapter 4.

1.4 Thesis Outline

The remainder of this thesis is structured as follows:

- In Chapter 2, we explore previous studies about motivation and incentives of users in crowdsourcing. Also, we present studies about online learning techniques which we use to solve the ISP. Specifically, participants in crowdsourcing projects have some similarities to employees in companies, members in organisations, or users in social networks. Thus, we first examine human motivation at work, incentive mechanisms in organisations, and user motivation and incentives in social networks and in crowdsourcing. We summarise the related works in a map to visually present the incentives, system characteristics associated with user motivation, dominant user behaviours, and the connections between them. After that, we present various models for multi-armed bandits that can be considered for dealing with both the ISP1 and ISP2. We then present Bayesian optimisation, a promising technique to deal with the ISP2.

- In Chapter 3, we formalise the ISP, our model to deal with the incentive problem. We also briefly introduce the ISP1 and ISP2, the two variants of the ISP that apply in different crowdsourcing projects (depending on the amount of the project's budget and prior knowledge about user performance in the project). Following this, we discuss how to measure the utility of the requester and the effectiveness of the incentives.
- In Chapter 4, we detail how do deal with the ISP1. In the chapter, we first formalise the ISP1. We then introduce a novel algorithm (**H AIS**) to solve the problem. Finally, we conduct extensive simulations to empirically evaluate the performance of **H AIS**. This helps us achieve the research objectives **RO1**, **RO2**, and **RO3**.
- In Chapter 5, we solve the ISP2. Similar to Chapter 4, the chapter begins by formalising the ISP2. Following this, we propose a novel algorithm (**B OIS**) to deal with the problem. After that, we assess the performance of the algorithm using extensive simulations. The chapter helps us attain the first three research objectives (**RO1**, **RO2**, and **RO3**). Also, together with Chapter 4, it helps us accomplish the last research objective (**RO4**).
- In Chapter 6, we summarise the accomplishment of this research. We also outline how this research can be improved upon in the future within the chapter.

Chapter 2

Literature Review

In this chapter, we provide an overview of previous research against which our work is positioned. To do so, we first present studies which are related to the user motivation and incentives in microtask crowdsourcing in Section 2.1. In this section, we organise many characteristics of crowdsourcing projects that might affect user motivation. We also review various incentives from existing studies that can be used to align user behaviours in crowdsourcing projects. Next, in Section 2.2, we present background literature which forms the basis of our approaches to the incentive problem. Specifically, the section describes studies about online learning approaches (multi-armed bandits and Bayesian Optimisation) that we use to build algorithms to deal with the incentive problem. Finally, in Section 2.3, we discuss which parts of the existing research form the point of departure for where new work is needed in order to fulfil the requirements outlined in Section 1.2.

2.1 Motivation and Incentives in Microtask Crowdsourcing

Behaviours of people are directed by their motivation. Thus, to align user behaviours in crowdsourcing projects, requesters should find ways to enhance their motivation in performing tasks or in participating in other activities. As will be shown later in the section, much research from many fields of study (such as psychology, sociology, economics, and computer science) shows that requesters can do this by providing appropriate incentives. Therefore, in Subsections 2.1.1 and 2.1.2, we present a review of previous studies about user motivation and incentives in microtask crowdsourcing. After that, in Subsection 2.1.3, we review studies about possible interactions between incentives and motivation.

2.1.1 Motivation

There are many well-established theories about the motivation of human in general, workers in companies, members in groups or organisations, or users in crowdsourcing. Based on these theories, we have identified dominant characteristics of crowdsourcing projects that have significant influence on user motivation. More details of the theories are presented in Appendix A. These characteristics are grouped into three categories corresponding to three sources of the motivation: *crowd* (related to size, structure and interaction of the worker community), *task* (related to designing tasks) and *system* (related to the platform and system goals). We now review the literature following these three categories.

First, regarding the crowd, there are many theories in social psychology and sociology learnt about motivations of people in groups or social communities such as social comparison theory (Festinger, 1954), common identity theory, common bond theory (Prentice et al., 2006), legitimate peripheral participation (Lave and Wenger, 1991), and collective effort model (Karau and Williams, 1993). All of these theories are described in Appendix A. Many studies have applied these theories in the context of crowdsourcing in order to explain several aspects of the motivation or provide approaches that can motivate users in crowdsourcing projects based on the relationships with others in the projects. For example, Ren et al. (2007) use common identity theory and common bond theory to create design principles (in online communities) about group (or community) size and off-topics in discussion forums that can develop the attachments of individuals to the group (or the community). While, Ling et al. (2005) use collective effort model (CEM) to build experiments to find what factors can motivate the contributions of users when working collectively in online communities. As predicted by CEM, their results show that emphasising the uniqueness of the contributions of individuals in the group will motivate their contributions.

Second, regarding the tasks, flow theory (Nakamura and Csikszentmihalyi, 2002) and the job characteristic model (Hackman and Oldham, 1980) are two well-established theories in psychology and organisational behaviour which can be used to design tasks that can motivate users in crowdsourcing systems. Nguyen et al. (2015) adopt flow theory in the context of crowdsourcing. They argue that flow¹ experience can lead to engagement of users. However, based on flow theory, three external conditions usually related to the present of the flow experience are ability-challenge balance (the balance between user ability and the challenge of the activity), clear goal, and immediate and unambiguous feedback. Hence, they conclude that these three conditions can indirectly affect user engagement in crowdsourcing systems. In contrast, Zheng et al. (2011) apply job characteristic model to identify what factors affect user participation in crowdsourcing contests. After developing a research model and analysing data from 283 users, they

¹See “Flow Theory” in Appendix A.

found that intrinsic motivations have more influence on participation of users than extrinsic motivations. And, three characteristics of the tasks that are positively linked with intrinsic motivations are autonomy (the freedom and control that users have when performing tasks), skill variety (the requirement of a wide range of skills and a variety of activities when performing tasks), and task complexity (the difficulty when performing tasks).

And finally, regarding the system, goal setting and user interface design are two dominant elements related to user motivation. Goal-setting theory (Locke and Latham, 2002) is a widely used theory to learn about human motivation. Based on the premise that incentives are affected by conscious goals, it identifies two core characteristics of goals that affect motivation of people in performing tasks. They are goal specificity (goals should be specific and measurable) and goal difficulty (the difficulty should be high enough to encourage effort and low enough to be achievable) (Locke and Latham, 2002). Ling et al. (2005) confirm these two characteristics in the context of online communities. On the other hand, much research has studied about what characteristics of user interfaces affect user motivation. Usability (the ease of use) and visual appeal (attractiveness) are two characteristics that are widely discussed (Holzinger, 2005; Hartmann et al., 2007; Phillips and Chaparro, 2009). Whilst, Malone (1982) has different approach in designing an enjoyable user interface. Malone (1981) found three factors that make computer games fun, which are *challenge* (goals with uncertain outcomes), *fantasy* (“mental images” that users evoke when interacting with the system), and *curiosity*. Based on these three factor, the author proposes a design framework to support these three factors. An example for a principle in the framework is to ask about the present of a clear goal in an activity. Inspired by Malone’s work, Brandtner et al. (2014) introduce design principles for the user interfaces of crowdsourcing systems. For example, they adopt Malone’s “clear goal” principle as a specific question “Is a clear goal visible for the user?”.

Besides the above-mentioned approaches that focus on specific characteristics of the system or motivation aspects of users, some other studies have looked at several characteristics or motivations to have a broader view about the user motivation in crowdsourcing. Indeed, Tokarchuk et al. (2012) propose a framework that designers can use to analyse incentive issue of crowdsourcing systems based on several characteristics of the systems. After the analysis, they can have a direction to identify appropriate incentives. However, the framework does not provide specific incentives to choose. And, it does not consider user behaviours, hence designers cannot use the framework to deal with particular behaviours of the users. Similarly, Kaufmann et al. (2011) introduce a model about user motivations in crowdsourcing grounded on self-determination theory and the job characteristic model. The model helps classify motivations of users in five categories (enjoyment-based motivation, community-based motivation, immediate payoffs, delayed payoffs, and social motivation) separated by two main branches (intrinsic motivation and extrinsic motivation). The model can be used to compare user motivations of a

system with others; however, in the view of designers, they cannot apply this model in the design process because it focuses only on user motivations without any connections with user behaviours and incentives.

In this section, we have introduced dominant works about user motivation. In the next section, we will review studies about incentives that might affect user motivation so that we can use to encourage users to perform tasks or participate in other activities in a crowdsourcing project.

2.1.2 Incentives

In the literature, there is a variety of incentives² that can be taken to motivate human in general, workers in firms or users in online communities in particular. We have collected these incentives from many fields of study including economics, psychology, social psychology, sociology and organisational behaviour. These incentives are grouped into five categories: *members* (related to individual workers or groups of them), *tasks* (related to designing and delivering tasks), *evaluations* (related to assessment methods of user performance), *rewards & punishments* (related to rewarding or punishing based on user activities) and *platform* (related to features provided by the system). Now, we present studies about incentives based on these categories.

Regarding the first category, members, many studies take advantage of the interactions between the members to take incentives that can encourage their participation, enhance their commitment, or improve their performance. Because people have tendency to compare with others and then change their behaviours towards the better (Festinger, 1954), we can provide environments for users to show their personal information and performance history to others. This incentive does not only help increase user performance (Huang and Fu, 2013) but also establish both bond-based and identity-based attachments (attachments to individuals and to groups as a whole) (Kraut et al., 2011). Additionally, we can make use of the collaboration between members to enhance their connections and improve the performance by providing communication channels. For example, in the citizen science project Fold It³ (<https://fold.it>), members can communicate with each other in three different ways: real-time talk through an Internet relay chat window, discuss by posting messages on a forum, or direct contact by private messages. This helps newcomers overcome the difficulties in the first time, helps players share their strategies or solutions to the puzzles, and importantly increases the motivations for them to continue playing the game (Curtis, 2015). Besides that, obtaining feedback from other members is also a useful method that can be used in crowdsourcing. According to feedback intervention theory (Kluger and DeNisi, 1996), after

²For a simple presentation, in this chapter when it does not lead to confusion we use “incentives” instead of “incentive methods”.

³An online video game that players, in each puzzle, find the most stable configurations of proteins by twisting, stretching or rearranging protein backbones and side chains.

receiving feedback, users can improve task performance or enhance motivations depending on many factors such as contents of the feedback, task characteristics or recipients' personalities. Based on this theory, [Zhu et al. \(2013\)](#) conduct a field experiment on Wikipedia (www.wikipedia.org) to test the effects of four different types of feedback: negative feedback (regulating recipients by warnings or reprimands), directive feedback (directing recipients by instructions, commands, etc.), positive feedback (energising recipients by work acknowledgement or rewards), and social feedback (maintaining social relationships, supporting group cohesion or self-confidence, etc.). The results, which agree with the predictions of the theory, suggest a trade-off that negative and directive feedback lead to improvement in task performance, while positive and social feedback lead to enhancement of general motivations. That means, system designers can consider using appropriate types of feedback to focus on task performance or general motivation.

Regarding the second category, tasks, an incentive that has started to attract the interest is personalised task recommendation (providing tasks based on users' interest and capabilities) ([Geiger and Schader, 2014](#); [Yuen et al., 2015](#); [Aldhahri et al., 2015](#)). The reason is that recommendation systems have been highly successful in many other areas, such as product recommendation (recommend products based customers' interest) or content personalisation (show appropriate contents to each user based on their interest) ([Geiger and Schader, 2014](#)). And if this approach is also successfully applied in crowdsourcing, it not only helps users find tasks faster and keep their motivations (because they do not take much time to find and choose the right tasks) but also helps requesters receive results with better quality (because the tasks are performed by the right users) and quicker. However, because there are some differences, applying the approach in crowdsourcing systems needs further research. Several studies have tried to point out difficulties ([Yuen et al., 2015](#)) and potential methods that can be used in crowdsourcing context ([Aldhahri et al., 2015](#)).

After delivering tasks and receiving results, we need to have appropriate incentives to evaluate user performance (the basis for giving rewards or punishments later). Evaluation is very important to user motivation and user performance. Indeed, if the evaluation is unfair among users, the ones that are under-evaluated will decrease their motivations. Additionally, imprecise evaluation can result in lower performance when contributions are over-evaluated (because they do not have to work as expected by requesters in order to be paid) or lower motivation when they are under-evaluated. Therefore, choosing appropriate incentives in the third category is important. There are several methods of evaluation in the literature ([Scekic et al., 2013](#); [Prendergast, 1999](#)), but they can be categorised by two ways. On one hand, based on the person who is in charge of the evaluation (computer or human), when contributions can be measured precisely, we can use objective evaluation (computer can do this); in other cases, we should use subjective evaluation ([Scekic et al., 2013](#)). On the other hand, based on the object of the evaluation, we can use individual-based evaluation when contributions of individuals can be easily

evaluated (objectively or subjectively), relative evaluation when setting absolute performance conditions is difficult or impossible, or team-based evaluation when it is difficult to evaluate contributions of individuals in a team (Scekic et al., 2013). Each method has its own benefits and side effects, thus choosing appropriate methods should be based on specific cases. For example, relative evaluation has positive effect on user performance because people tend to compare with others around and try to improve themselves towards the better people (Festinger, 1954). However, it can diminish the motivation to help others. Similarly, team-based evaluation can support the collaboration between members in a team, but it can lead to free-riding phenomenon (Prendergast, 1999).

After evaluating user contributions, we need to reward or punish them based on the evaluations. Similar to the evaluation, compensation is attached to user motivation and also user performance. Regarding this fourth category, there are various incentives that can be used to reward users. According to Zichermann and Cunningham (2011), there are four types of rewards: Status, Access, Power and Stuff (SAPS). *Status* is the relative position of a person in relation to others in the system. We can use badges, levels, or leader boards to show status. *Access* is the opportunity to interact with private resources or interact in a special way, such as being able to see the performance history of other members or to see some information before others. *Power* is special rights of users over others, for example can deactivate or activate accounts of other members. And *stuff* means tangible rewards such as money or anything else that can be converted easily into money.

Although this classification (SAPS) is used in gamification (the practise of applying game elements in non-game contexts), it covers most types of rewards that can be used. They are in order of decreasing the desire, the complexity in usage, but increasing in the price. Indeed, for example, using status requires us to take more time to think about its usage. Basically, in order to use status, the first step is to choose fundamental elements (can be points). The next step is to build rules to earn points and then some more rules to identify point thresholds to increase levels. And then some more rules to deliver badges for levels. However, by using for example money, it is extremely easy but the most expensive.

In crowdsourcing, a widely used type of reward is financial rewards (stuff). Although it is easy to use money to reward, it is complicated to use in order to have optimal effects on user motivation and user performance. For example, we often believe that increasing payment leads to improving quality. However, Mason and Watts (2010) and Araujo (2013) show that higher payment does not result in higher performance but more users (join a task). One reason for this, according to Mason and Watts (2010), is the presence of an “anchoring effect”, the effect occurs when people get higher payment, they will perceive that the value of their contribution is greater, so that they do not put more effort. In another case, Shaw et al. (2011) find that when using financial incentives properly (used in combination with asking users to guess the responses of others on the

same tasks), it can produce results with higher quality. Additionally, one of the most complicated and controversial issues related to tangible rewarding is its relation with intrinsic motivation of users. We will discuss this in more detail in Section 2.2.1

Regarding the last category, the platform, one incentive can be used to enhance user motivation and performance is improving the usability (the ease of use) and the visual appeal (attractiveness) of the user interface. In general, we can consider using any principles to improve the usability and the appeal of the user interface. However, in order to know how well the interface is designed, we need to know how to evaluate a website or an application with respect to these two characteristics, such as [Holzinger \(2005\)](#) (to evaluate the usability) or [Hartmann et al. \(2008\)](#) (to evaluate the usability and the appeal). In particular, as described in Section 2.1.1, if the application tends to have an enjoyable user interface, designers can consider using directly the framework developed by [Malone \(1982\)](#) which focuses on three factors: challenge, fantasy and curiosity, or using its extension in crowdsourcing context ([Brandtner et al., 2014](#)).

Besides the above-mentioned incentives, contests are shown to be an efficient approach in crowdsourcing, however, they do not belong to a specific category. Specifically, they use relative evaluation (evaluations category), tangible rewarding (rewards & punishments category), and time pressure (tasks category). 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 multiple contests with single prize⁴ ([DiPalantino and Vojnovic, 2009](#)) or a single contest with multiple prizes⁵ ([Luo et al., 2016, 2015](#); [Cavallo and Jain, 2013](#); [Chawla et al., 2012](#); [Cavallo and Jain, 2012](#); [Archak and Sundararajan, 2009](#); [Moldovanu and Sela, 2001](#)). The prize distribution can be a certain number (one, i.e., winner-take-all, or more) of the best-performance (or luckiest) users ([Luo et al., 2016, 2015](#); [Cavallo and Jain, 2013](#); [Chawla et al., 2012](#); [Archak and Sundararajan, 2009](#); [Moldovanu and Sela, 2001](#)), or all the users who make non-zero contributions ([Cavallo and Jain, 2012](#)). The design objective can be maximising the principal's benefit ([Archak and Sundararajan, 2009](#); [Cavallo and Jain, 2013](#); [Chawla et al., 2012](#); [DiPalantino and Vojnovic, 2009](#); [Moldovanu and Sela, 2001](#)) the agent's utility, or the social welfare ([Cavallo and Jain, 2012](#); [Luo et al., 2015](#)). In terms of principal's benefit, a number of researches focus on (to maximise) the whole submissions ([Luo et al., 2016, 2015](#)) or only on the submissions of the highest-performance agents ([Chawla et al., 2012](#); [Archak and Sundararajan, 2009](#); [DiPalantino and Vojnovic, 2009](#)). To model the contests for analyses, a popular approach in contest design studies

⁴Multiple-contest with single-prize is the model where the users face more than one contests at a time and have to choose which one(s) to compete with others for a prize. This model is used in some contest platforms that crowdsource solutions for problems (e.g., designing logo for a company) such as Taskcn.

⁵This model is better for the incentive selection problem in microtask crowdsourcing as it is simpler by omitting the contest-choosing stage but more flexible with the prize distribution (can be single or multiple prizes).

is using all-pay auctions⁶ as it reflects the fact in crowdsourcing that every participant of a contest has to exert a certain level of effort (Luo et al., 2016; Chawla et al., 2012; DiPalantino and Vojnovic, 2009; Moldovanu and Sela, 2001). Besides, most recently, Luo et al. (2015) take a different approach by using Tullock contests⁷ as they relax the level of competition by spreading the opportunities to win the prizes for all agents (who exert non-zero contributions). This approach can be effective in crowdsourcing projects as it might attract more users to the contests.

Although the work on the contest design is plenty, applying this body of research in building efficient contests for real-world crowdsourcing projects is still challenging because of the following reasons. First, these studies are based on the rationality assumption⁸, whereas real users in crowdsourcing might be partly rational or irrational, as they might lack information, knowledge, or time. Second, the studies do not consider other factors related to the users' intrinsic motivation that might affect their behaviour, such as the project purpose, the task nature, or the participant community as described earlier. Third, contests in these studies are mainly for creative tasks (e.g., designing a company logo or finding solutions for a problem) where it takes a lot of time (can be hours or even days) for performing. Because of these reasons, applying the results of this body of research into microtask crowdsourcing might be different. To deal with this, depending on a specific microtask crowdsourcing project, these studies can be used together with some other studies in psychology, sociology, or computer science (e.g., Frey and Jegen (2001); Gneezy and Rustichini (2000); Heyman and Ariely (2004); Mason and Watts (2010); Rogstadius et al. (2011)) for better understanding the 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 intrinsic and extrinsic motivations of the users in order to build more appropriate prize structures.

We have reviewed many incentives can be used to motivate users. The effects of these incentives are not clearly understood, for example the effects of tangible rewards on intrinsic motivation. Hence, in order to use them effectively, we need to learn more about possible interactions between these incentives and user motivation.

2.1.3 Interactions between Incentives and Motivation

Although the incentives are numerous (as presented in the previous subsections), the effectiveness of particular incentives in specific crowdsourcing projects is usually unknown in advance. In other words, the net-outcome of these incentives is hard to identify. In

⁶All-pay auctions are the auctions where every bidder (i.e., agent) has to pay for their bid (i.e., exert effort).

⁷Tullock contest is a contest where the probability that agent $i = 1, 2, \dots$ (who exerts effort b_i) wins the contests is $p_i = b_i^r / \sum_j b_j^r$ where $r > 0$. The simplest form of Tullock contests are *lotteries* where $r = 1$.

⁸The assumption is that the human users act in a fully rational manner.

some cases, the incentives result in the effect which is higher (i.e., user performance is better) than expected. Yet, in some other cases, the resulted effect is lower. According to numerous social studies, this is because, in some cases, the incentives enhance (which is called *crowd in*) the intrinsic motivation of users to perform tasks. Thus, they have a positive effect on user performance. And in some other cases, the incentives diminish (which is called *crowd out*) users' intrinsic motivation, and hence they have a negative effect on user performance. These crowding effects reflect the fact that there exist possible interactions between incentives.

Research about crowding effects has a long history for over 45 years (Cerasoli et al., 2014). However, the relationship between external interventions and on intrinsic motivation to perform an activity is still an open challenge. In psychology, Deci (1971) conduct laboratory and field experiments to examine this relationship. They conclude that monetary incentives may reduce intrinsic motivation for conducting an activity. Also, verbal and positive feedbacks tend to increase such motivation. After that, a lot of studies in social sciences (both theoretically and empirically) have been conducted to investigate these effects. A meta-analysis of 129 studies is conducted by Deci et al. (1999). They conclude that, in general, tangible rewards have negative effects on intrinsic work motivation. And, these effects are stronger in more interesting activities. Conversely, in an another meta-analysis, Cameron et al. (2001) come up with a different conclusion, that in general, rewards do not diminish intrinsic motivation. In particular, rewards enhance intrinsic motivation when the activities are of low interest. These two studies have one point in common that negative effects happen on high-interest activities. Recently, Cerasoli et al. (2014) conduct a more thorough meta-analysis which covers hundreds of studies about crowding effects for over 40 years. They conclude that the net outcome of incentives and intrinsic motivation has a strong impact on user performance. However, the conditions in which the impact is positive are still needed to study.

In economy, researchers and practitioners often believe in the relative price effect, which is higher payment causes higher performance (Frey and Jegen, 2001). That means the performance level is proportional to the payment. They assume the intrinsic motivation is constant, and hence it is often omitted. Yet, when conducting a survey about offering incentives in firms, Prendergast (1999) concede the possibility that intrinsic motivation might be affected by external factors. Then, Frey and Jegen (2001) combine the psychologists' idea of crowding effects and the economists' relative price effect into a model. Figure 2.1 illustrates the model in the case the crowding-out effect happens. In their model, work effort is proportional to the compensation level, as presented in the figure as the solid line S (which is called *supply curve* in economical studies). But in some cases, external factors (such as feedback or motivational messages) might crowd in or crowd out intrinsic work motivation. They integrate these crowding effects by shifting the supply curve downward (or upward) a distance according to how large the crowding-out (or crowding-in) effect is. As shown in the figure, when external factors undermine

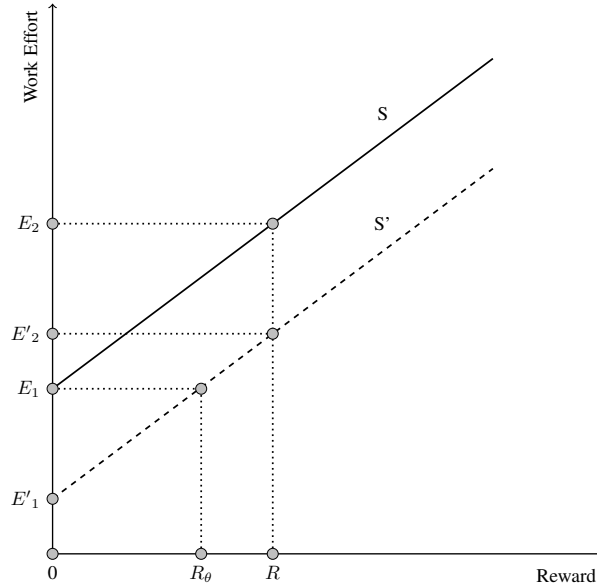


FIGURE 2.1: Net-outcome of the Price- and the Crowding-Out-Effect (Frey and Jegen, 2001). S , the solid line, presents the price effect and S' , the dashed line, presents the price effect when intrinsic motivation is crowded out by external factors. E_1 and E_2 are the work efforts of users corresponding to rewards of 0 and R respectively when intrinsic work motivation is not affected by external factors. E'_1 and E'_2 are the work efforts of users corresponding to rewards of 0 and R respectively when intrinsic work motivation is affected by external factors. R_θ is the reward level at which the corresponding work effort when external factors undermine intrinsic motivation is the same as that when the reward is 0 and external factors do not affect intrinsic motivation.

intrinsic motivation, the supply curve is shifted downward. Thus, the work effort is reduced from E_1E_2 down to $E'_1E'_2$ when the reward ranges from 0 to R . This figure shows two things about the combination of the relative price effect and the crowding-out effect. The first one is that even when intrinsic motivation is crowded out by external factors, work effort is still proportional to the reward. The second one is that when users perceive their autonomy in performing activities is negatively affected by the factors, their intrinsic work motivation is diminished and hence their work effort is dropped from E_1 to E'_1 (if there is no reward). In this case, the reward must be large enough in order to encourage users' work effort to the same level as when the reward is not present. In the figure, the work effort when the reward is R_θ (if external factors undermine intrinsic motivation) is the same as when the reward is 0 (if external factors do not undermine intrinsic motivation).

The model proposed by Frey and Jegen (2001) is consistent with what was found by Gneezy and Rustichini (2000). Gneezy and Rustichini (2000) study how different compensation levels affect user performance. After conducting two field experiments on Amazon Mechanical Turk, they found that a higher compensation level results in a higher performance of users (i.e., they complete more tasks). An interesting point in their findings is that there exists a discontinuity in user performance when a compensation is offered. Concretely, when users are paid for their tasks with a small reward,

their performance drops significantly compared to their performance when no reward is offered. This drop in user performance is indeed the shift of the supply curve in the model of [Frey and Jegen \(2001\)](#). This is the reason the authors suggest “pay enough or don’t pay at all” as the main message which is presented as the title of the paper. In their experiments, the tasks are interesting (IQ test questions) or meaningful (collecting monetary donations).

Another study, conducted by [Heyman and Ariely \(2004\)](#), can help us explain why user effort drops significantly when external factors start to affect user intrinsic motivation. This study investigates the cause-and-effect relationship between financial rewards and work effort. They propose a model in which the factor that decides the work effort of users is the type of the market whether it is monetary, social, or mixed. Based on the results of three field experiments, they conclude that in monetary markets, where the expected rewards relate to money (e.g., £1.00), higher payment results in higher level of effort. Whereas, in social markets, where the expected rewards do not relate to money (e.g., a £1.00 chocolate bar when its price is not mentioned), different payment levels do not yield different levels of effort. And, in mixed markets, where both elements of monetary and social markets are present (e.g., a £1.00 chocolate bar with its price mentioned), they tend toward monetary markets. That means the effort of users is decided mainly by how the users perceive the corresponding market as monetary or social.

The above-mentioned studies investigate crowding effects in classroom activities, volunteer works, or other non-crowdsourcing environments. [Rogstadius et al. \(2011\)](#) is a rare study that examines interactions between extrinsic incentives and intrinsic motivation and how these affect user performance in crowdsourcing. More specifically, they learn how financial rewards affect task quality (how good the tasks completed), task quantity (number of tasks completed), task completion time (how quickly the tasks are completed), and task attraction (the extent to which the tasks attract users to perform). Their experiments, run on Amazon Mechanical Turk, show that higher rewards result in a higher rate in attracting users, more tasks completed, quicker task completion, but not better task quality. The results from the experiments also show that intrinsic motivation (emphasising the meaningfulness of the tasks) helps improve the quality of the tasks. This quality improvement occurs when extrinsic incentives are low. They also suggest further research to better understand possible interactions between incentives and user performance in crowdsourcing projects.

To sum up, although researchers have identified important aspects of the effect of external factors on user performance via their intrinsic motivation, it needs more work to better understand this relationship as suggested by [Frey and Jegen \(2001\)](#), [Rogstadius et al. \(2011\)](#), [Cerasoli et al. \(2014\)](#), and many other authors. This motivates us, in this thesis, to find a more practical way to identify the most effective incentive for a particular crowdsourcing project. In the next section, we will present our approach that uses

machine learning techniques to learn the most effective incentive among the incentives suggested by social studies.

2.2 Online Learning

As mentioned above, in a specific crowdsourcing project, we are not sure which incentive is the best and how to implement an incentive that can maximise the requester's utility. Thus, our approach is to use machine learning algorithms to learn the most effective incentive. In this section, we briefly present the approach and review the studies in which our approach is based on. We first describe the incentive selection problem (ISP) in Subsection 2.2.1. Next, we review studies about two methods that we use to solve the ISP. Specifically, we discuss the literature about MABs in Subsection 2.2.2 and Bayesian Optimisation in Subsection 2.2.3.

2.2.1 Incentive Selection Problem

Although there are plenty of incentives that can be used to incentivise users in crowdsourcing, choosing the most appropriate incentives in a specific crowdsourcing project is still challenging. Therefore, one practical way for dealing with the incentive problem (i.e., providing appropriate incentives) is to design incentives that are likely to be effective based on previous studies and then empirically select the most effective one. More specifically, the above-mentioned studies can be used to design several incentive methods. An example is that, using pay for performance (an incentive method) with the base payment (a parameter) ranges from £1.00 to £2.00 and the payment for every 10 tasks completed (another parameter) is £0.20. Another example is that, using contests (another incentive method) with the group size ranges from 5 to 30 and the amount of prize money for the best user is from £1.00 to £5.00. Other relevant studies in psychology, sociology, or computer science, e.g., Frey and Jegen (2001); Gneezy and Rustichini (2000); Heyman and Ariely (2004); Mason and Watts (2010); Rogstadius et al. (2011), can also be used for this task, as they help us better understand possible interactions between the factors related to motivations of the users (e.g., the incentives, the level of autonomy and interestingness of the tasks, or the purpose of the projects). Then, based on the proposed incentive methods, an adaptive approach could be used to identify the most effective efficiently, where an incentive is an incentive method with specific values of the parameters. We refer to this as the incentive selection problem (ISP).

2.2.2 Multi-armed Bandits

Following this direction, as noted earlier, multi-armed bandits (MABs) form a promising approach. In the classical stochastic MAB problem (Robbins, 1952), the agent is given

a slot machine with I arms; at each time step, the agent pulls an arm and receives a reward which is drawn from an unknown fixed probability distribution given the arm. The agent's goal is to maximise the cumulative sum of rewards. If the agent knows the expected reward corresponding to each arm, it will just pull the best arm, i.e., the one which has the highest expected reward. Nonetheless, the agent does not know these expected rewards beforehand. So, it has to sample the arms to estimate these values (exploration) in order to utilise the best one (exploitation). In other words, the agent needs to balance between exploration and exploitation in order to achieve the goal. In the ISP, an arm corresponds to an incentive, and pulling an arm means offering the corresponding incentive to a group of users. The reward of pulling an arm is the total utility that the agent has from the users in the group. To be easier to review related studies and connect these studies with the ISP, we can formulate the MAB problem as follows. The slot machine has a finite collection of arms $i \in \{1, \dots, I\}$, where the expected reward of arm i is μ_i which is unknown a priori. At time step t , the agent pulls an arm (i) several ($n_i^{(t)}$) times and receives a reward (a utility of $r_i^{(t)}$). The objective is to maximise the expected utility after T time steps:

$$\max \sum_{t=1}^T n_i^{(t)} \mu_i. \quad (2.1)$$

The ISP has several characteristics that we should consider in order to choose appropriate variations of the classical MABs to deal with the ISP effectively. First, the ISP has budget constraints (both financial and time budgets). Second, the pullings are in batches (i.e., an arm can be pulled multiple times and multiple arms can be pulled at a time). Third, the ISP has group-based nature of the arms which is discussed in Section 1.1. Fourth, in some cases of the ISP, there might be many candidate incentives (i.e., arms) in which the incentives related to an incentive method might have possible correlations. In these cases, the financial budget might be not enough to learn all the arms. There are many variations of the classical MABs have been investigated. Some of them which have one or some of the above-mentioned characteristics which can be considered to use in dealing with the ISP. Next, we will review the studies about MABs which are related to these characteristics in Subsections 2.2.2.1 and 2.2.2.2. The studies presented in these two subsections are not conducted in the context of crowdsourcing. Thus, in Subsection 2.2.2.3 we will discuss some other studies that attempt to solve the incentive problem by using MAB techniques. After that, in Subsection 2.2.2.4, we will present a variation of MABs that deal with many arms.

2.2.2.1 Budget Constraints

In the ISP, pulling an arm incurs a pulling cost (e.g., £2) and the agent has a limited budget for the pullings (e.g., £500). Formally, it costs c_i to pull arm i once. Also, the

agent has a budget of B for the pullings and it has to finish after T time steps. A number of studies considering budgeted MABs have been conducted. The (financial) budgeted MAB model was first introduced by [Tran-Thanh et al. \(2010\)](#), 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. [Sen et al. \(2015\)](#) 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 chain⁹ show these algorithms are effective, especially `SOAAv` with its adaptive approach.

Taking a different approach, [Tran-Thanh et al. \(2012\)](#) use the idea of upper-confidence bounds (UCBs) from [Auer et al. \(2002a\)](#) to build an algorithm called `Fractional KUBE` (henceforth, `fKUBE`), that combines exploration and exploitation in one process. Specifically, it spends the first I time steps (from 1 to I) to pull each arm once in order to have initial estimates of the arms. Then, at time step t ($t > I$), it pulls the arm that maximises $\text{ucb}_i^{(t)}$, where $\text{ucb}_i^{(t)} = \hat{\mu}_i/c_i + \sqrt{(2 \ln t)/n_i^{(t)}/c_i}$ is the UCB of the estimate of arm i at time step t . By choosing the arms like this, `fKUBE` integrates further exploration into the exploitation phase. Actually, as the estimates are uncertain, instead of looking at the estimate of an arm 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 ($\sqrt{(2 \ln t)/n_i^{(t)}/c_i}$). In more detail, when an arm is pulled, this square root term (representing the uncertainty of this arm's estimate) will decrease. Therefore, regarding this term, the arms which are applied less (hence, are more uncertain) have more opportunity to be pulled in the next period¹⁰. We also apply this idea of UCBs to our algorithms (presented in Chapters 4 and 5) for having efficient strategies in identifying the best incentive. The work of [Badanidiyuru et al. \(2018\)](#) approaches budgeted MABs more generally by dealing with multi-dimensional bandits (each dimension corresponds to a resource, such as financial budget or time budget). More specifically, in their model (which is called Bandit with Knapsacks), there are d resources and each resource (i) has a limited budget (B_i) to be consumed. Pulling an arm incurs costs on these d resources and receives a reward. The objective is to maximise the overall reward within the predefined budgets. The authors propose two algorithms (`PD-BwK` and `BalanceBwK`). However, these algorithms cannot be applied to the ISP because the resources in their model cannot be shared, whereas in the ISP, the time (a resource in their model) can be shared, i.e., pulling one or more

⁹In 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.

¹⁰See [Sutton and Barto \(2018\)](#) for more discussion on UCBs.

arms several times (providing one or more incentives to several groups) can happen at a given round.

Nonetheless, as we will show in Section 4.3, the algorithms developed by [Sen et al. \(2015\)](#) and [Tran-Thanh et al. \(2010, 2012\)](#) are not efficient when dealing with the ISP. It is because these algorithms do not consider the group-based nature of the incentives (i.e., arms) and the time budget. Also, even the algorithms are designed to work with the financial budget, they do not have a good exploration-exploitation balance. More precisely, they do not have an effective and adaptive mechanism for distributing the financial budget across the time steps. For instance, with ϵ -first, we have to specify an appropriate value of ϵ in advance. Yet, it is difficult to do this when there is little information about the performance of users in a specific project. And when differences in the effectiveness of the arms are low (i.e., it is difficult to differentiate the arms' 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 when possible) to not only evaluate their performance but also to benchmark our new algorithms.

2.2.2.2 Batched Pulling

In the ISP, the pullings are in batches. That means, in each round multiple incentives can be offered and an incentive can be offered to different user groups. Various studies about batched MABs have been conducted. [Anantharam et al. \(1987\)](#) extend the classical MAB model of [Lai and Robbins \(1985\)](#) from single play into multiple plays. Concretely, in their model, at each time step the arms have to be pulled in batches in which each batch has K out of I arms. The rewards corresponding to a specific arm are i.i.d. In this model, sizes of the batches are fixed. When considering online combinatorial optimisation, [Audibert et al. \(2014\)](#) propose a more popular model which is called semi-bandit feedback. The model fits better the ISP as sizes of the batches can be different. Specifically, an incentive $a^{(t)} = (a_1^{(t)}, \dots, a_I^{(t)}) \in \mathcal{A} \subseteq \{0, 1\}^I$ conducted at time step t specifies whether an arm is pulled or not ($a_i^{(t)} = 0$ or 1). In these two models, each arm is pulled maximum once in each round. However, an incentive in the ISP can be offered to different groups of users at the same time. That means the arm corresponding to an incentive can be pulled multiple times in each round.

[Niculescu-Mizil \(2009\)](#) relax the number of times each arm can be pulled at a time step. In their model, at each time step, they assume to have K coins, each coin can be used to pull an arm, and several coins can be used for an arm. Nonetheless, they only consider the case where the rewards of pulling an arm at a time step are the same. Yet, in the ISP, the utilities received from different groups of the same incentive are different and

are assumed to be i.i.d. [Perchet et al. \(2016\)](#) introduce a more general model where sizes of the batches can be different and an arm can be pulled multiple times in a batch. However, the model is only for two arms. Recently, [Esfandiari et al. \(2019\)](#) and [Gao et al. \(2019\)](#) extend the model of [Perchet et al. \(2016\)](#) to many arms. Yet, as many other multiple-play MAB models, in the proposed model, exactly K out of I arms have to be pulled in each round. A more general model, compared to that of [Gao et al. \(2019\)](#), is proposed by [Jun et al. \(2016\)](#). They call it (b, r) -batch MABs where in a batch of b pulls, an arm is pulled maximum r ($1 \leq r \leq b$) times. We can see that $(1, 1)$ -batch MAB is the classical MAB. Though, as they investigate the pure exploration problem, they do not consider costs and budget constraints.

Several other works have considered MABs with both batched pulling and budget constraints such as [Luedtke et al. \(2019\)](#), [Xia et al. \(2016\)](#), and [Zhou and Tomlin \(2018\)](#). Yet, in their models, the agent has to play a fixed number, K , of the arms ($1 \leq K \leq I$) and each arm is played at most once in each round. Moreover, they do not examine the time budget as in the ISP. As we can see from the above discussion, there is much work about batched MABs. However, each study focuses on a different aspect of the batched pulling as in the ISP. Hence, their proposed algorithms cannot be applied for the ISP. Because of this, we need to build different mechanisms that can consider all the aspects of the batched pulling together with the budget constraints and the group-based nature of the incentives so as to have efficient pulling policies.

2.2.2.3 Crowdsourcing

The above-mentioned studies are about MABs in general. There are not many studies that try to use MABs to solve the incentive problem in crowdsourcing. [Itoh and Matsubara \(2016\)](#) investigate whether using MAB techniques to deal with the incentive problem is effective or not. Concretely, in their model, there are T tasks that need to be completed in T time steps (one task at a time step). At each time step t , the requester provides task t together with one of I ($I < T$) predefined incentives and receives a utility which is drawn from a normal distribution with unknown mean and variance values corresponding to the offered incentive. The objective is to maximise the requester's overall utility. However, their model is rather simple and difficult to use in practice. In more detail, they do not consider batched pulling. So, when there are a lot of tasks, providing the tasks one by one will take a lot of time to finish. This is likely the case in microtask crowdsourcing as the tasks are usually small and hence numerous. Also, they do not consider the budget constraint. Specifically, pulling an arm in the model (i.e., offering the corresponding incentive) does not incur a cost. Additionally, they do not consider the group-based nature of the incentives. Therefore, their model cannot be used to deal with the ISP.

Taking a different approach, [Ho et al. \(2016\)](#) consider the incentive problem as a combination of MABs and the classical principal-agent model. More specifically, they formalise the incentive problem as a multiple-round process. In each round, one user (i.e., worker) completes a task based on a contract designed in advance by the requester. From the performance of the users 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 ([Kaufmann et al., 2011](#)). In these projects, the outcomes might not be proportional to payments ([Heyman and Ariely, 2004](#); [Rogstadius et al., 2011](#)). 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.

2.2.2.4 Many-armed Bandits

Thus far in the discussion, we have reviewed the works about MABs where the number of arms is smaller than the number of possible pulls. However, in some cases of the ISP, the number of candidate incentives is large compared to the budget. So, the above-mentioned algorithms cannot be used in these cases, as it is impossible for them to pull all the arms to learn the best one. Many-armed bandits are introduced to deal with these cases ([Berry et al., 1997](#); [Li and Xia, 2017](#); [Wang et al., 2008](#)). [Berry et al. \(1997\)](#) first proposed simple strategies to solve the infinitely *many-armed bandits*. Yet, their work is focusing on Bernoulli arms where the reward of pulling an arm is either 0 or 1. Then, [Wang et al. \(2008\)](#) extend to a more general reward distribution. After that, [Li and Xia \(2017\)](#) embed budget constraints to the model. However, these studies do not consider possible correlations between the arms as in the ISP. Thus, they are not efficient for the ISP. For example, to avoid sampling all the arms (which is impossible), the algorithm proposed by [Li and Xia \(2017\)](#) is to choose randomly K arms and then apply an UCB-based algorithm for these arms. The value of K is calculated so that it can balance between choosing enough arms to explore and choosing not too many arms which can take up a large portion of the budget. But, when the arms have correlations in terms of the reward distributions, choosing the arms like this does not make use of the correlations to identify a better arm.

Therefore, another variation of MABs that takes advantage of this information, *continuum-armed bandits*, is a better candidate for the ISP. [Kleinberg et al. \(2008\)](#) study the Lipschitz MAB problem where the mean-reward distribution is assumed to be smooth and the smoothness is identified by a constant which is called Lipschitz constant. [Trovo](#)

[et al. \(2016\)](#) extend the work of [Kleinberg et al. \(2008\)](#) to cover budget constraints. However, the algorithms proposed by these two studies (**Zooming** and **B-Zoom**) require to know the Lipschitz constant in advance. Yet, in a specific crowdsourcing project, we usually do not have such information, which is the user performance, beforehand. Taking a different approach to deal with continuum-armed bandits, [Bubeck et al. \(2011\)](#) use tree-based optimisation to build an algorithm (**H00**) which can result in more effective pulling policy. Specifically, the idea of **H00** is that it uses a binary tree to gradually partition the search space (i.e., the arm set) into smaller parts. The root node covers the whole search space. Each node has two child nodes which cover two separate parts of the node. **H00** assigns an optimistic estimate of the max utility of the arms corresponding to each node. Based on this tree, in each round, the algorithm explores the arms by randomly pulling an arm in the part corresponding to the leaf which has the maximum estimate. After receiving the utility of this arm, it will update the estimates of nodes on the path from the root to the corresponding leaf. By doing like this, **H00** can focus exploration on parts which have high probabilities of containing good arms. More recently, [Mattos et al. \(2018\)](#) introduce another algorithm (**LG-H00**), which is based on **H00**, to be used in a real world application. The purpose of using **LG-H00** is to identify appropriate times to send notifications to users in a company to help the users have positive experiences since sending too early or too late might result in negative impacts on the user experience. Although **LG-H00** and **H00** algorithms have a good exploration strategy and do not need a prior knowledge about the smoothness of the mean-reward function as in **Zooming** and **B-Zoom**, they cannot be used for the ISP when one of the chosen incentive methods has more than one parameter. This is because the basic data structure of these two algorithms is a binary tree which is used for identifying the optimal point in one-dimensional objective functions by partitioning the search space into parts. However, in the ISP, the number of dimensions of the mean-reward function of an incentive is the number of parameters of the incentive. Hence, in order for the algorithms to be used for the ISP, they need to have another data structure that can work on multidimensional objective functions. Also, the algorithms are designed for extremely irregular mean-reward functions (the smoothness is determined by the Lipschitz constant), whereas the functions in crowdsourcing are unlikely to be complicated. For example, changing the prize for the best user slightly from £3.00 to £3.01 is unlikely to change the utility a lot. Therefore, in order to solve the ISP effectively, it needs to find another approach to take more advantage of the possible correlations between the arms. So in the next subsection, we will be discussing a more promising technique for this problem, which is Bayesian Optimisation, since it can take more advantage of the correlations to learn the best incentive quickly.

2.2.3 Bayesian Optimisation

Bayesian Optimisation (BO) has emerged as an efficient approach to optimising a black-box function where evaluations are expensive (e.g., taking a substantial amount of time to conduct an evaluation or incurs an economic cost at each evaluation). In fact, BO has been used in various applications. For example, in AlphaGo (Silver et al., 2016), a computer programme that plays the game of Go, the authors had to tune many hyper-parameters (Chen et al., 2018). To evaluate the effectiveness of a single hyper-parameter value, it takes about 6.7 hours even when they run multiple games in parallel by using 400 GPUs¹¹. And if they use grid search¹² to optimise 6 hyper-parameters (5 possible values per hyper-parameter), they need about 8.3 days. After applying BO to tuning AlphaGo’s hyper-parameters, they not only reduce the amount of time spent on the tuning but also improve the win-rate of AlphaGo significantly (Chen et al., 2018). Another example is in circuit optimisation. Actually, in the circuit design process, designers need to run simulations to evaluate a design. Yet, a simulation can take hours, days, or even weeks to complete (Lyu et al., 2018; Okobiah et al., 2014). So, BO attracts more interest in this field of study; that is to find the best design by using as few simulations as possible (Lyu et al., 2018; Zhang et al., 2019). Besides, BO has been used in many other applications such as speed recognition (Dahl et al., 2013), image classification (Snoek et al., 2015), robotics (Calandra et al., 2014), retrieval systems (Li and Kanoulas, 2018), recommender systems (Galuzzi et al., 2020), and machine learning algorithms (Snoek et al., 2012).

Formally, we consider the problem of finding a global maximiser (or minimiser) of an unknown objective function f :

$$\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \quad (2.2)$$

where \mathcal{X} is usually a compact subset of \mathbb{R}^d and f is continuous. Also, the shape of f is unknown in advance, and the only way to estimate this function is through evaluation. In more detail, we can choose some positions, $\mathbf{x} \in \mathcal{X}$, to evaluate and then observe the corresponding values of the function, $y = f(\mathbf{x})$. The observed values can be perturbed by stochastic noise. The idea of BO is to use a probabilistic model, which is called a *surrogate model* and denoted by \mathcal{M} , to estimate the objective function. After observing more data (i.e., samples of the objective function), the model is refined via Bayesian posterior updating. The posterior embodies our updated beliefs about the objective

¹¹A GPU (graphic processing unit) is a single-chip processor which is initially designed to handle intensive graphics tasks. However, because of larger memory and capability in internal parallelism, GPUs are becoming more popular in other purposes, especially running machine learning algorithms (Steinkraus et al., 2005; Tavara, 2019)

¹²Grid search is a simple way to find the best values of the hyper-parameters by trying all possible combinations of the hyper-parameter values (Bergstra and Bengio, 2012).

Algorithm 1 Bayesian Optimisation

Input: f , T , and n_0 \triangleright the unknown costly objective function, the time horizon, and the number of samples identified by a space-filling design

Output: \mathbf{x}^* \triangleright a point at which the objective function is maximised

01: $\mathcal{D}_0 \leftarrow \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_0}$; \triangleright initial dataset includes samples from a space-filling design
02: $\mathcal{M}_0 \leftarrow$ update the model; \triangleright calculate posterior mean & standard deviation ($\mu(\cdot)$ & $\sigma(\cdot)$)
03: **for** $t = 1, \dots, T$ **do**
04: $\mathbf{x}_{n_t} \leftarrow \operatorname{argmax}_{\mathbf{x}} \alpha(\mathbf{x}, \mathcal{D}_{t-1})$; \triangleright select a new point by optimising the acquisition function; $n_t = n_0 + t$
05: $y_{n_t} \leftarrow f(\mathbf{x}_{n_t})$; \triangleright query the objective function to obtain the output
06: $\mathcal{D}_t \leftarrow \{\mathcal{D}_{t-1}, (\mathbf{x}_{n_t}, y_{n_t})\}$; \triangleright add the new sample to the dataset
07: $\mathcal{M}_t \leftarrow$ update the model;
08: $\mathbf{x}^* \leftarrow \operatorname{argmax}_{\mathbf{x}} \mu(\mathbf{x})$; \triangleright identify the final point at which the posterior mean is maximised
09: **return** \mathbf{x}^* ;

(Frazier, 2018; Li and Kanoulas, 2018; Shahriari et al., 2016)

function. Based on the posterior, BO then uses an *acquisition function*, $\alpha : \mathcal{X} \mapsto \mathbb{R}$, to guide exploration, i.e., choosing the next point to evaluate. Specifically, the acquisition function incorporates the uncertainty of the posterior and the posterior itself. This function has a high value at a position if the position has a high uncertainty about the estimate. The function also has a high value at the position whose estimated value is high. Thus, the acquisition function represents the trade-off between exploration (which tends to sample high uncertain points) and exploitation (which tends to sample high estimated points). BO will determine the point which has the maximum value in the acquisition function to evaluate next.

The procedure of BO is shown in Algorithm 1. An illustrative example of how BO works over three time periods is shown in Figure 2.2. In the figure, the point to be chosen to sample (i.e., new observation) in the third period ($t = 3$) is at the position that the acquisition function (evaluated at the end of the second period) has the maximum value. In the same manner, after observing the actual value of this point and then updating the posterior and the acquisition function, the updated acquisition function will guide the point to be sampled in the next period ($t = 4$).

As presented above, an important component of BO is the surrogate model to model the objective function. The de facto surrogate model which is often used popularly in BO is Gaussian processes (Chen et al., 2018; Frazier, 2018; Lyu et al., 2018; Snoek et al., 2012). Another important component is the acquisition function. In the following subsections, we will discuss in detail these two components.

2.2.3.1 Gaussian Process Prior

A Gaussian Process (GP) (Rasmussen and Williams, 2006) is a powerful non-parametric statistical model over functions. A GP not only helps us estimate the shape of the

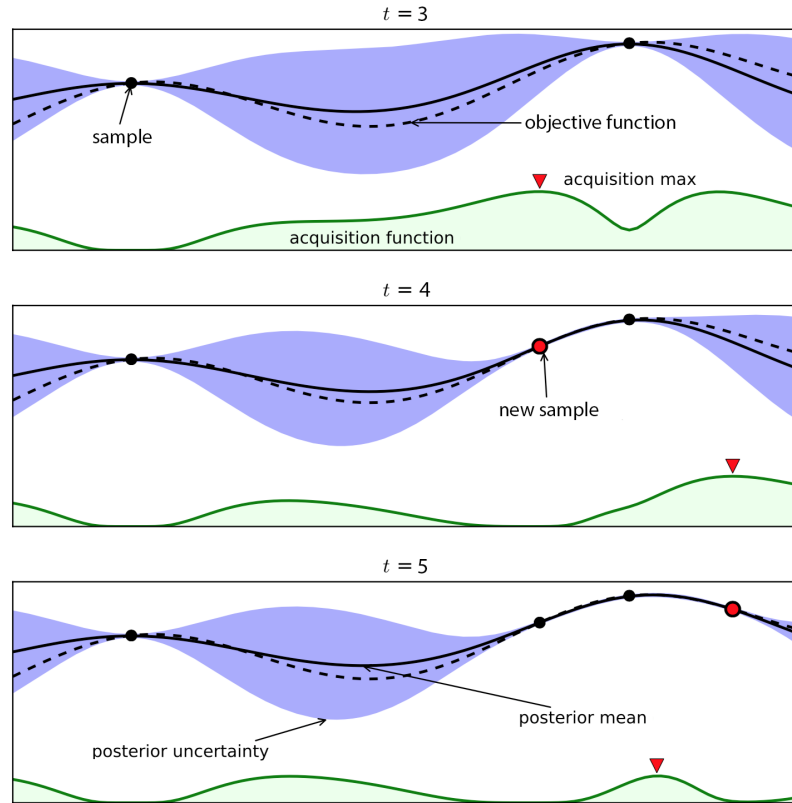


FIGURE 2.2: An illustrative example of how Bayesian Optimisation works over three periods (Brochu et al., 2010). In each plot corresponding to a period, the objective function $f(\cdot)$ (the dashed line, which is unknown) is shown together with the posterior mean $\mu_t(\cdot)$ (the solid line) and the posterior uncertainty $\mu_t(\cdot) \pm \sigma_t(\cdot)$ (the purple area around the posterior mean). The green shaded plot at the bottom is the acquisition function $\alpha(\cdot)$.

objective function (in the form of the posterior mean $\mu(\cdot)$) but also provides us the uncertainty of the estimate (in the form of the posterior standard deviation $\sigma(\cdot)$). A GP is a distribution over functions, which can be completely determined by a prior mean function $\mu_0(\mathbf{x}) : \mathcal{X} \mapsto \mathbb{R}$ and a positive definite covariance function (a.k.a., *kernel*) $\kappa(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu_0(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')), \quad (2.3)$$

where

$$\mu_0(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \text{ and} \quad (2.4)$$

$$\kappa(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \mu_0(\mathbf{x}))(f(\mathbf{x}') - \mu_0(\mathbf{x}'))]. \quad (2.5)$$

If $f \sim \mathcal{GP}(\mu_0, \kappa)$ then $f(\mathbf{x}) \sim \mathcal{N}(\mu_0(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x})) \forall \mathbf{x} \in \mathcal{X}$, i.e., $f(\mathbf{x})$ is Gaussian. In a noisy environment, we do not observe the function values, $f(\mathbf{x})$, but only their noisy versions, $y = f(\mathbf{x}) + \epsilon_{\text{noise}}$. Here, the noise is assumed to be i.i.d normal, i.e., $\epsilon_{\text{noise}} = \mathcal{N}(0, \sigma_{\text{noise}}^2)$. At time step t , based on the observations so far, $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_t}$ (where $n_t = n_0 + t$), the model, \mathcal{GP}_t , is fitted by its posterior mean and variance functions. Specifically, for any point \mathbf{x} , we have:

$$\mu_t(\mathbf{x}) = \mathbf{k}_t(\mathbf{x})^T (\mathbf{K}_t + \sigma_{\text{noise}}^2 \mathbf{I})^{-1} \mathbf{y} \quad \text{and} \quad (2.6)$$

$$\sigma_t^2(\mathbf{x}) = \kappa(\mathbf{x}, \mathbf{x}) - \mathbf{k}_t(\mathbf{x})^T (\mathbf{K}_t + \sigma_{\text{noise}}^2 \mathbf{I})^{-1} \mathbf{k}_t(\mathbf{x}), \quad (2.7)$$

where \mathbf{K} is the square matrix whose values are $\mathbf{K}_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{k}_t(\mathbf{x})$ is a $(n_t \times 1)$ vector whose i th value is $\kappa(\mathbf{x}_i, \mathbf{x})$, and \mathbf{y} is the vector whose i th value is y_i .

We have presented GP prior which acts as the surrogate model of BO. However, the model is updated at every time step after we receive a new sample of the function f to represent our belief about f . In Subsection 5.2.1, when discussing about how to apply BO into solving the ISP2, we will discuss in more detail the choice of covariance function. The next issue is how to effectively choose a new point to explore. BO deals with this by using an acquisition function which we will discuss in the next subsection.

2.2.3.2 Acquisition Functions

There are many acquisition functions to be used in the literature and in real world. Yet, we will present the ones which can be considered to use in the ISP.

Probability of Improvement

The idea is to maximise the probability of improvement (PI) over a target τ :

$$\begin{aligned} \alpha_{\text{PI}}(\mathbf{x}; \mathcal{D}_t) &\stackrel{\text{def}}{=} P(f(\mathbf{x}) \geq \tau) \\ &= \Phi(\gamma(\mathbf{x}; \mathcal{D}_t)), \end{aligned} \quad (2.8)$$

where

$$\gamma(\mathbf{x}; \mathcal{D}_t) = \frac{\tau - \mu_t(\mathbf{x})}{\sigma_t(\mathbf{x})}. \quad (2.9)$$

Here, $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. The idea was first introduced by [Kushner \(1964\)](#). This approach can perform very well when the target is known ([Shahriari et al., 2016](#)). However, in general this value

is unknown. In these cases, the performance of probability of improvement depends significantly on the choice of the target τ . Indeed, if τ is too small, the search tends to be around local optima. But if τ is too large, although the search can jump quickly towards the global optimum, it is slow to reach this optimum as it does not have fine-tune steps towards the optimum (Jones, 2001). One option is to choose τ as the current best value of the observations, $y^+ = \max_{i=1:n_t} y_i$. Another heuristic choice of τ can be the best posterior mean based on the observations so far. Yet, as presented above the probability of improvement tends to exploit intensively towards the points which have high probability of being better than τ (Brochu et al., 2010; Shahriari et al., 2016). Specifically, as shown in Equation 2.8, the approach only considers if there is an improvement, i.e., if the output $f(\mathbf{x})$ is greater than or equal to the target τ . It does not consider the magnitude of the improvement, i.e., $f(\mathbf{x}) - \tau$.

Expected Improvement

One possible way to overcome the pure exploitation as in the probability of improvement is to measure the expected improvement (EI) (Mockus et al., 1978). Literally, we incorporate also the magnitude of the improvement. Formally, the EI function is defined as below:

$$\begin{aligned} \alpha_{\text{EI}}(\mathbf{x}; \mathcal{D}_t) &\stackrel{\text{def}}{=} \mathbb{E}[\max\{f(\mathbf{x}) - \tau, 0\}] \\ &= (\mu_t(\mathbf{x}) - \tau)\Phi(\gamma(\mathbf{x}; \mathcal{D})) + \sigma_t(\mathbf{x})\phi(\gamma(\mathbf{x}; \mathcal{D})), \end{aligned} \quad (2.10)$$

where $\phi(\cdot)$ is the probability density function of the standard normal distribution. As shown in Equation 2.10, EI tries not only to find a better point than τ (i.e., exploitation) but also to make use of the uncertainty obtained after updating the surrogate model to conduct exploration.

GP Upper Confidence Bound

GP Upper confidence bound (GP-UCB), which is based on the idea of UCB in MABs, is another way to balance between exploration and exploitation (Srinivas et al., 2010). Formally, a GP-UCB acquisition function is of the following form:

$$\alpha_{\text{UCB}}(\mathbf{x}; \mathcal{D}_t) = \mu_t(\mathbf{x}) + \lambda\sigma_t(\mathbf{x}). \quad (2.11)$$

In Equation 2.11, λ is a tunable parameter to adjust the exploration-exploitation balance. One possible advantage of using GP-UCBs as acquisition functions is that UCBs are inherently designed to minimise the cumulative regret over the course of optimisation. This seems to be better when dealing with the ISP.

Thompson Sampling

Thompson sampling (TS), proposed by [Thompson \(1933\)](#), has been shown experimentally and theoretically to be an efficient approach for sequential decision making under uncertainty ([Agrawal and Goyal, 2012](#); [Chapelle and Li, 2011](#)). Also, TS has been shown to be an efficient acquisition function when it is applied to BO ([Basu and Ghosh, 2018](#); [Kandasamy et al., 2018](#)). The idea of TS is based on randomisation. Concretely, at a time step t , it first samples from the posterior. That means,

$$\alpha_{\text{TS}}(\mathbf{x}; \mathcal{D}_t) \sim \mathcal{GP}_{t-1}, \quad (2.12)$$

where \mathcal{GP}_{t-1} is the GP’s posterior updated at the previous time step, $t - 1$. Then, as with other acquisition functions it chooses the next point by maximising α_{TS} .

We will come back to these acquisition functions later when we apply BO to the ISP. Specifically, in [Chapter 5](#) we will discuss which functions to be used when dealing with the ISP and the reasons for the choice.

2.3 Summary

This chapter provides the background knowledge and the detailed review of the relevant literature. In particular, we first presented studies about incentives, user motivation, and interactions between incentives and user motivation in microtask crowdsourcing. We then reviewed studies related to our approach to the incentive problem which is to use machine learning techniques to learn the best incentive in a specific crowdsourcing project.

Regarding the user motivation and incentives, our conclusions are as follows. First, there are numerous studies about user motivation and incentives that cover many important aspects of the population. In more detail, dominant aspects of the population were examined, from user participation to user persistence and user performance. Specifically, there is a large number of established works about user motivation. Important characteristics of crowdsourcing systems attached to user motivation were investigated in the literature. Similarly, previous work has investigated a variety of incentives that can be used in many cases. Second, these studies come from many fields of study (including psychology, sociology, economics, and computer science) and each one focuses on specific issues (such as task design, goal setting, attachments to group, teamwork, and task recommendation). Last, it is challenging to understand user motivation and having effective incentives to encourage users in a specific microtask crowdsourcing project. More specifically, several studies have investigated approaches to retain users as long as possible in a work session. However, it needs further studies to make these approaches more

practical. Similarly, although a large body of research about motivation and incentives related to engagement of human in general, of workers in companies, of members in virtual communities, or of contributors in crowdsourcing, there is a lack of research to find specific ways to encourage users performing tasks on a regular basis. Additionally, most of the studies focus on separate incentives, while combinations of them can result in better (or worse) effects.

Therefore, one practical way to solve the incentive problem in microtask crowdsourcing is to solve the incentive selection problem (ISP). Concretely, instead of trying to identify the best incentive in a specific crowdsourcing project (which is very difficult, or almost impossible, in many cases), we choose several good candidate incentives and then learn the best one. Regarding this, we then presented studies about online learning which can help us design algorithms to deal with the ISP effectively. In more detail, we reviewed two approaches, multi-armed bandits (MABs) and Bayesian optimisation (BO). Various MAB models have been proposed. Yet, as the ISP is complex and has multiple characteristics (i.e., batched pulling, financial and time budget constraints, and group-based nature of the arms), no models consider all of the characteristics. However, some ideas of the corresponding algorithms can be used for the ISP. In particular, BO forms a promising approach for the ISP when there are many candidate incentives. In the next chapter, we will present the ISP together with its two variants. Then, in the following two chapters we will propose our solutions to the two variants of the ISP.

Chapter 3

The Incentive Selection Problem

This chapter forms the basic to the next two chapters (Chapters 4 and 5) which help us achieve the research objectives discussed in Section 1.2. Specifically, in Section 3.1 we explicitly present the incentive selection problem (ISP) in the form of a batched 2d-budgeted group-based MAB problem. Then, in Section 3.2, we differentiate the three variants of the ISP (ISP1, ISP2, and ISP3). In Chapters 4 and 5, we will go into details together with our solutions to these variants. After that, in Section 3.3, we discuss how to measure the utility of the requester in a specific crowdsourcing project. This is the basic to the algorithms which helps not only define the learning objective and but also evaluate the effectiveness of the algorithms. Finally, in Section 3.4 we introduce how the proposed algorithms and the benchmarks measure the effectiveness of the incentives. Based on this measurement, each algorithm can develop its own strategy to effectively learn the best incentive.

3.1 The Problem

Suppose a requester wants to run a crowdsourcing project. The objective is typically to maximise the requester’s cumulative *utility* with a given financial budget B and time budget T . For ease of presentation, in some places of the thesis when it does not lead to confusion between the two types of budget, we use “budget” for the financial one. The *utility* of the requester on a task is the benefit that the requester receives after the task is completed which can be measured by including task quantity, task quality, task completion time, or some subset of them. We will be discussing how to measure the utility of the requester in Section 3.3. To achieve the above-mentioned objective, the requester spends the available budget on providing incentives to encourage participants (referred to as *users*) to perform tasks. We refer to this as the *incentive problem*. As discussed in Section 2.1.2, although there are various incentive methods to motivate users, it is still an open challenge in identifying the best incentive method in a specific

crowdsourcing project. Therefore, a more practical way to solve the incentive problem is to choose some candidate incentive methods which can hopefully have positive effects on the performance of users. Then, a good strategy should be applied to select the best incentive method together with the best implementation of the method (i.e., the optimal values of the method's parameters) in order to maximise the cumulative utility. The candidate methods can be chosen from previous studies about user motivation and incentives or from experience. The incentive methods can be in the form of contests or individual-based (i.e., non-contests). Yet, to keep the presentation simple, we consider individual-based incentive methods as contests in which the group size (the number of users in a contest) is one. Hence for now, when we use “incentive methods”¹, it means they are in the form of “contests”. Also in this work, we only consider the contests where the submissions of all users (not only the best users) in a contest are useful to the requester. In other words, the utility is *additive*. This assumption prevents the model to be applied in projects where the requesters only consider the best submissions in every contest. Yet, this normally happens in macrotask crowdsourcing. For example in macrotask crowdsourcing systems 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, every task completed is typically useful to the requesters (Feyisetan and Simperl, 2019; Ramchurn et al., 2013). Thus, the assumption is reasonable in the research of this thesis.

The incentive methods can be different in their *parameters*. These differences can be presented as the difference in the parameter set and the difference in the values of the parameters. A parameter of an incentive method is a variable that we can change in order to adjust the implementation of the method. For example, the group size can be considered as a parameter whose values are in a specific range, e.g., $\{1, 2, \dots, 100\}$. Or, the motivation method can also be a parameter whose values can be 1 for emphasising the meaningfulness of the work (Chandler and Kapelner, 2013), 2 for using entertaining diversions (Dai et al., 2015), or 3 for arousing the curiosity of users (Law et al., 2016)². Other possible parameters of an incentive method can be the performance evaluation method (e.g., the number of tasks completed or the quality of the tasks submitted), or the motivation method (e.g., showing motivational messages or using entertaining diversions). We assume that there are *correlations* between the incentives in an incentive method. The correlation between two incentives in a method is the extent to which difference in effectiveness of the incentives based on the values of their parameters. In more detail, with each incentive method, when changing slightly the value of a parameter the effectiveness of the method is supposed to change slightly. Hence, an incentive method is referred to as an incentive cluster (or *cluster* for short). That means a cluster is a group of incentives corresponding to an incentive method having the same

¹Sometimes just use *incentives* (instead of *incentive methods*) when it does not lead to confusion

²See Section 2.1.2 for more information about these and many other motivation methods.

parameter set but different values of the parameters. Examples of possible correlations between the incentives in a cluster are shown in Figure 1.1³.

Since the effectiveness of the incentives is usually unknown in advance, we are interested in finding an efficient means of selecting incentives (i.e., exploring their effectiveness and then exploiting the most effective one) to maximise the requester’s overall utility. We refer to this as the *ISP*.

3.2 Three Variants of the ISP

On a specific crowdsourcing project, we can have a different strategy to solve the ISP effectively. So, we consider three variants of the ISP which correspond to three different types of crowdsourcing projects. For a simpler presentation, we refer to the cluster in which its incentives are correlated as *correlated cluster* and the cluster in which its incentives are not correlated as *uncorrelated cluster*. The first variant (uncorrelated ISP or simply ISP1) corresponds to the projects where all clusters are uncorrelated. The second variant (correlated ISP or simply ISP2) corresponds to the projects where all clusters are correlated. And the third variant (mixed-correlated ISP or simply ISP3) corresponds to the projects where some clusters are correlated while the others are not. We can see that these three variants cover all possible crowdsourcing projects. In the following subsections, we detail these three variants.

3.2.1 The ISP1 (Uncorrelated ISP)

This is the variant where the incentives are not correlated. This corresponds to crowdsourcing projects with a small financial budget or when we have prior knowledge about user performance. In more detail, on some projects the financial budget is not large enough⁴ to learn the best incentive among a large number of candidate incentives in the chosen clusters. Thus, we have to continue choosing some incentives (among various candidate incentives in an incentive method) in which we think they are better than the others. The number of chosen incentives should be small enough⁴ to estimate all the incentives within the given budget, i.e., the remaining budget (after exploring the incentives) should be large enough to exploit the best incentive explored. On some other projects we might have prior knowledge about the performance of users. And hence, we can continue choosing a small number of incentives (among various candidate incentives in an incentive method) which are good enough in incentivising the users. An example of the chosen incentives in this variant is shown in Figure 1.1. More specifically, the chosen ones might be a_1 and a_2 in Figure 1.1a (a cluster) and b_1 , b_2 , and b_3 in Figure 1.1b

³See Section 1.1 for more discussion about the correlations shown in this figure.

⁴Details of how to choose an appropriate number of incentives based on their parameters and the given budget will be discussed in Subsection 4.3.4.

(another cluster). That means the final candidate incentives (after choosing the second time) are not correlated. hence, we refer to this variant of the ISP as the *uncorrelated ISP* or *ISP1*.

3.2.2 The ISP2 (Correlated ISP)

This is the variant where the incentives in each cluster are correlated. This corresponds to crowdsourcing projects with a large financial budget and when we do not have good prior knowledge about user performance. In more detail, on some projects the financial budget is large enough to learn the best incentive while keeping all candidate incentives in the chosen clusters. The budget might not be large enough to apply each incentive at least once to estimate its effectiveness, it is large enough to have an efficient strategy in applying the incentives so as to maximise the overall utility of the requesters by making use of the correlations between the incentives. Also, on these projects we do not have any prior knowledge about the performance of users. So, we cannot continue choosing some incentives among various incentives in each method. However, as we a large enough budget we can keep all the candidate incentives in all incentive methods so that we do not miss the best incentives. That means the candidate incentives in each cluster are correlated. Thus, we refer to this variant of the ISP as the *correlated ISP* or *ISP2*.

3.2.3 The ISP3 (Mixed-correlated ISP)

This is the variant where the incentives in some clusters are correlated while the incentives in the other clusters are not. This corresponds to crowdsourcing projects where after choosing some candidate incentive methods we continue choosing some candidate incentives in *some* incentive methods. That means, with some incentive methods, the chosen incentives are uncorrelated as we only kept some incentives; however, with the other methods, the chosen incentives are correlated as we kept all incentives. This might be because we have some prior knowledge about the performance of users in these projects. So, it helps us be confident in choosing some incentives in several incentive methods, but not the other methods. We refer to this variant of the ISP as the *mixed-correlated ISP* or *ISP3*.

To deal with the ISP3, we have two ways. The *direct* way is to build an algorithm to solve it directly. The *indirect* way is to apply the solution to the ISP1 or the ISP2 for it. Specifically, to apply the solution to the ISP1 to the ISP3, we need to continue choosing some incentives in correlated clusters. Also, to apply the solution to the ISP2 to the ISP3, we simply consider each incentive in uncorrelated clusters as the only incentive in a cluster. In this thesis, we chose this indirect way to solve the ISP3 as we can make use of the solutions designed for the other variants (ISP1 and ISP2). Although this way

might not be the best way for the ISP3, it is practically acceptable if the solutions to the other two variants are effective. The direct way is left for future work.

3.3 Measuring the Utility of the Requesters

In order to choose an appropriate metric for the utility of the requester in a crowdsourcing project, we need to know the purpose of the project. So, in the ISP we only consider projects with a *specific* and *fixed* purpose. This is reasonable as in many cases we can convert variable purposes into several specific and fixed ones. Indeed, in practice there can be long-lasting crowdsourcing projects in which over their lifetime there can be different dominant aspects of the population to deal with. Accordingly, the requesters can consider using various incentives to focus on enhancing specific aspects at different stages. For example, in the early stages of development, incentives need to be chosen in order to result in the fast recruitment of sufficient numbers of users. After that, the incentives can be changed to focus on retaining existing users in a work session as long as possible. Finally, as the project matures, small changes can be made to existing incentives to emphasise the quality of the work completed by users or the across-session persistence (i.e., encourage users to come back to the project to perform tasks on a regular basis) of users. With these long-lasting projects, each stage can be considered as a sub-project with its own purpose (i.e., focusing on specific aspects) and budgets. For a simple presentation, in the rest of the thesis we use the term “project” to refer to a crowdsourcing project with a specific purpose and hence the utility metric is fixed during its lifetime. In other words, in a project, the requester can include one or some of the above-mentioned aspects in the utility function. Based on these chosen aspects, the requester can then choose appropriate incentive methods for the selection process so as to maximise the overall utility.

In a crowdsourcing project, depending on its purpose we can include task quantity, task quality, task completion time, or some subset of them in the utility metric. For example, [Yin and Chen \(2015\)](#) consider the quantity and quality of the tasks. In detail, the utility function in their study is assumed to be $r = w_h n_h + w_l n_l - c n_{\text{all}}$ where w_h (w_l) is the utility obtained from a high (low) quality incentivised task, n_h (n_l) is the number of high (low) quality incentivised tasks, n_{all} ($= n_h + n_l$) is the total number of incentivised tasks, and c is the economic cost of the investigated incentive (i.e., using bonuses) for an incentivised task. Here, an incentivised task is a task in which the bonus, the incentive used, occurs. A note when choosing an aspect to be included in the metric is that it should not only measure the effectiveness of the incentives appropriately, but also be possible to evaluate the effectiveness easily and ideally in an automatic manner. That is because after offering an incentive in a period of time, the effectiveness of this incentive should be calculated quickly before deploying another ones (to another groups of users) in the next period. For example, in [Ramchurn et al. \(2013\)](#), the tasks are

drawing an evacuation route from a building to the nearest road 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 using an aggregation method such as majority voting or expectation maximisation (Nguyen et al., 2013). In detail, we can ask some other users 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.

3.4 Measuring the Effectiveness of the Incentives

To measure the effectiveness of the incentives, we use *density*, i.e., the utility-cost ratio (Tran-Thanh et al., 2010), as it reflects the expected utility (i.e., reward in the context of MABs) obtained per cost unit. The density of incentive ι is defined as

$$\delta_\iota = \frac{\mu_\iota}{c_\iota}, \quad (3.1)$$

where μ_ι is the expected utility of incentive ι and c_ι is the cost of applying this incentive once. However, as the real densities of the incentives are unknown in advance, we have to estimate them. Right after period t , the estimate of the density of incentive ι is:

$$d_\iota^{(t)} = \frac{\hat{\mu}_\iota^{(t)}}{c_\iota}. \quad (3.2)$$

In this equation, $\hat{\mu}_\iota^{(t)} = (1/m_\iota^{(t)}) \sum_{j=1}^t r_\iota^{(j)}$ is the current estimate of incentive ι 's expected utility, where:

- $m_\iota^{(t)} = \sum_{j=1}^t n_\iota^{(j)}$ is the number of times incentive ι has been applied until the end of period t ,
- $n_\iota^{(j)}$ is the number of times incentive ι is applied in period j (i.e., incentive ι is offered to $n_\iota^{(j)}$ different groups)⁵, and
- $r_\iota^{(j)}$ is the total utility of applying incentive ι in period j (i.e., the total utility of users in all groups of incentive ι in period j).

⁵An illustrative example of this one is described in Section 4.1.

All algorithms examined in this work only estimate the expected utility of an incentive when the incentive has already been offered before. So, in Equation 3.2, $m_i^{(t)} > 0 \forall i$. In Chapters 4 and 5, during the learning process of an algorithm, we have to constantly update the estimates of incentives' effectiveness. Therefore, to keep the presentation simple, by default we use “the estimate of an incentive” instead of “the estimate of an incentive's density (or effectiveness)”. Also, we use “the best (worst) incentive” to denote the incentive with the highest (lowest) estimate, as opposed to “the real best (worst) incentive”.

Chapter 4

The ISP with Uncorrelated Candidate Incentives

In this chapter, we deal with the ISP1 (or uncorrelated ISP) that is a variant of the ISP in which the candidate incentives are not correlated. This is the variant when the budget for the crowdsourcing project is small in comparison to the costs of the candidate incentives. So, after choosing candidate incentive methods, we have to continue choosing some candidate incentives in each incentive method to keep the total number of candidate incentives small enough¹. Thus, after exploring, the remaining budget is large enough¹ to effectively exploit the best incentive explored. This is also the case when we have prior knowledge about the user performance in the project. Hence, in each cluster, we can choose some candidate incentives with a high confidence that they are good. In this chapter, we aim to build a solution to the ISP1 which can help us achieve **RO1** (the solution is efficient), **RO2** (the solution is implemented in an autonomous manner), **RO3** (the solution is adaptive), and part of **RO4** (the solution can be applied in a certain number of crowdsourcing projects, here the solution covers the projects where the candidate incentives are not correlated). We first formalise the ISP1 as a batched 2d-budgeted group-based MAB problem in Section 4.1. Then, we introduce **H AIS**, a novel adaptive algorithm to solve the ISP1 in Section 4.2. After that, we present an empirical evaluation of the algorithm by running simulations in Section 4.3. Finally, we conclude the chapter in Section 4.4.

4.1 The Problem

We have described a general description of the ISP in Chapter 3. Now, we formally present the first variant of the ISP, i.e., the ISP1, when we continue choosing some

¹Details of how to choose an appropriate number of incentives based on their structures and the given budget will be discussed in Subsection 4.3.4.

a fixed unknown distribution with mean 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 :

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

This model is Contribution 1 as presented in Section 1.3.

4.2 The HAIS Algorithm

As discussed in Section 2.2.2, although there are many MAB algorithms that can be considered to use for the ISP1, since the problem is complex (with several characteristics), these algorithms are not efficient to deal with the ISP1. Therefore, in this section we introduce **Hoeffding-based Adaptive Incentive Selection** (henceforth, **HAIS**), an adaptive algorithm for the ISP1. We first present the Hoeffding's inequality and how HAIS uses this inequality to adaptively build an applying policy (Subsection 4.2.1). After that, we give an overview of the algorithm (Subsection 4.2.2). Finally, we detail how HAIS is built and how it acts in the exploration (Subsections 4.2.3 and 4.2.4) and exploitation (Subsections 4.2.5 and 4.2.6) phases.

4.2.1 The Hoeffding's Inequality

In general, Hoeffding's inequality (Hoeffding, 1963) 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 the examples. In particular, let Y_1, \dots, 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 states that:

$$P(\bar{Y} - E[\bar{Y}] \geq \gamma) \leq \exp\left(\frac{-2n\gamma^2}{(y_{max} - y_{min})^2}\right) \quad \text{and} \quad (4.1)$$

$$P(\bar{Y} - E[\bar{Y}] \leq -\gamma) \leq \exp\left(\frac{-2n\gamma^2}{(y_{max} - y_{min})^2}\right), \quad (4.2)$$

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} . Applying this to the ISP1, we can determine the number of sampled users needed in particular incentives to obtain good estimates of the effectiveness of the incentives. And hence, we can identify the best incentive with a certain level of confidence.

We choose Hoeffding’s inequality as it helps us identify the appropriate numbers of times the incentives are applied in the exploration phase dynamically based on the estimates of the incentives’ effectiveness 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. In order to obtain initial estimates, the target number of sampled users in each incentive is at least 25. Thus, in the first periods, incentive 1 is applied 3 times and incentive 2 is applied 5 times. Then, when applying Hoeffding’s inequality, the result says that to obtain a certain confidence level of L_h (e.g., 50%) in being sure that the current best incentive is the real best one, each incentive needs to have 60 sampled users. 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 25 sampled users, we need to apply this incentive seven times to have 35 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 the real best incentive, or at least one of the most effective incentives. 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 do not have to choose different values in different crowdsourcing projects in order to obtain effective applying policies as is required with parameter ϵ in ϵ -first (Section 2.2.2.1). More specifically, we could choose a fixed value of 50% for L_h in all crowdsourcing projects. With each project, based on L_h and the estimates of the incentives so far, HAIS will adaptively identify the appropriate number of sampled users needed on each incentive. However, with ϵ -first, in projects where the financial budget is large, we should choose small values of ϵ , 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 , such as 0.10, to have sufficient budget to explore.

4.2.2 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.

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 U_1 (e.g., 20) users have been sampled for each incentive. In order to obtain significant estimates of the incentives, U_1 should be large enough to have a meaningful estimate of the incentive,

e.g., 20. 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 (corresponding to a certain level of confidence L_e , e.g., 95%) 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., $d_i^{upper} < d_j^{lower}$). Eliminated incentives will not be applied in the Hoeffding step.

In the second period, HAIS applies *Hoeffding's* inequality as described in Section 4.2.1. 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 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 ϵ_1 , e.g., 0.1). This budget bound for exploration (i.e., $\epsilon_1 B$) is applied to both 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 ϵ_1 as an upper bound for the budget for exploration. The parameter that helps HAIS identify the budget for exploration is L_h . 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 situations.

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 ϵ_2 (e.g., 0.5, to have two equal parts). 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, it uses the given budget ($\epsilon_2 b / T_s$) to apply the best incentive as many times as possible followed by an update to the estimate of this incentive. 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.

To illustrate the algorithm, Figure 4.2 shows an example of how HAIS acts in a simple case. In the first period of the example (day 1), incentives 1 and 3 are applied four times, while incentive 2 is applied only twice, to have enough $U_1 = 8$ users. Note that the numbers chosen in this example (e.g., U_1 or g_i) are for illustrative purposes only. After this period, suppose that the estimate of incentive 3 is significantly lower than that of incentive 1 (i.e., $d_3^{upper} < d_1^{lower}$). Incentive 3 is therefore eliminated. Hence,

| DAY | 1 | 2 | 3 | 4 | 5 |
|--------|-------------|-----------|----------------------|-----|-------------------|
| GROUPS | | | | | |
| COST | £24 | £8 | £12 | £12 | £24 |
| STEP | Sampling | Hoeffding | Stepped Exploitation | | Pure Exploitation |
| PHASE | Exploration | | Exploitation | | |
| | | | | | |

FIGURE 4.2: An example of running HAIS where $I = 3$, $g_1 = 2$, $g_2 = 4$, $g_3 = 2$, $T = 5$ (days), $B = £80$, $c_1 = £2$, $c_2 = £4$, $c_3 = £2$, $\epsilon_1 = 0.40$, $\epsilon_2 = 0.50$, $U_1 = 8$, incentive 1 is the real best incentive, incentive 3 is the real worst.

in the Hoeffding step conducted on day 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. After the exploration phase, we also suppose that the estimate of incentive 1 is higher than that of incentive 2. Thus, incentive 1 is applied in the third period (day 3), followed by an update to this incentive's estimate. In the next period (day 4), as the estimate of incentive 1 still appears to be higher, HAIS just applies this incentive with the given budget (£12). In the last period, it applies the best incentive (incentive 1) 12 times with the remaining budget (£24). A high-level overview of the algorithm is presented in Algorithm 2.

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. These help us achieve Contribution 2 as presented in Section 1.3. In the following subsections, details of the four steps will be discussed. The explanations will be linked to the corresponding parts of the detailed pseudocode of HAIS shown in Algorithm 3.

Algorithm 2 High-level Overview of the HAIS Algorithm

| | |
|---------------|---|
| | Input: financial budget (B), time budget (T), number of incentives (I), .. |
| | Predefined parameters: $\epsilon_1, \epsilon_2, U_1, L_h, ..$ |
| | Output: applying policy, overall utility, number of periods used |
| | Note: See Algorithm 3 for the full detail version of the HAIS algorithm. |
| Sampling | $\left\{ \begin{array}{l} 01: \text{Apply all incentives so that each incentive has about } U_1 \text{ sampled users.} \\ 02: \text{Update the estimates of all incentives, i.e., } d_i, d_i^{lower} \text{ and } d_i^{upper} \forall i = 1 \dots I. \\ 03: \text{Eliminate clearly ineffective incentives from being applied in the Hoeffding step. An} \\ \text{incentive } (i) \text{ is eliminated if there exists another incentive } (j) \text{ whose the lower bound of} \\ \text{the estimate } (d_j^{lower}) \text{ is larger than the upper bound of its estimate } (d_i^{upper}). \end{array} \right.$ |
| Hoeffding | $\left\{ \begin{array}{l} 04: \text{Calculate the target number of sampled users } (U_2) \text{ so as to identify the best incentive} \\ \text{with a confidence level of } L_h \text{ (see Section 4.2.1).} \\ 05: \text{Apply all active incentives so that each incentive has about } U_2 \text{ sampled users.} \\ 06: \text{Update the estimates of all active incentives.} \end{array} \right.$ |
| Stepped Expl. | $\left\{ \begin{array}{l} 07: \text{The budget for stepped exploitation is } b_2 = \epsilon_2 b, \text{ where } b \text{ is the residual budget.} \\ 08: \text{This budget } (b_2) \text{ is distributed across } T_s \text{ periods, where } T_s + 1 \text{ is the remaining periods} \\ \text{including the last one.} \\ 09: \text{In each of these } T_s \text{ periods, apply the best incentive as many times as possible and then} \\ \text{updates the estimate of this incentive.} \\ 10: \text{Note that this step might not use all } T_s \text{ periods (see Section 4.2.5).} \end{array} \right.$ |
| Pure Expl. | $\left\{ \begin{array}{l} 11: \text{Apply the best incentive with the residual budget.} \\ 12: \text{Apply the second best incentive with the residual budget.} \\ 13: \text{Repeat the process until the residual budget is not enough to apply any incentive.} \\ 14: \text{return the applying policy, the overall utility, and the number of periods used.} \end{array} \right.$ |

4.2.3 The Sampling Step

As discussed above, the objective of this step is twofold: to obtain initial estimates of the incentives' effectiveness and to preclude clearly ineffective incentives from being used in the next period.

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 is U_1 . 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 ($\sum_{i=1}^I c_i/g_i$). The purpose of the budget bound for exploration is to prevent spending too much budget on exploring. 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's density (Line 6) and an update on the confidence interval of the incentive's estimate (Line 7). The confidence interval of incentive i 's estimate (d_i^{lower}, d_i^{upper}) is:

$$d_i^{(1)} \pm z_e \frac{s_i^{(1)}}{\sqrt{n_i^{*(1)}}}. \quad (4.3)$$

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 period 1;
- $s_i^{(1)} = \frac{1}{c_i^*} \sqrt{\frac{\sum_{u=1}^{n_i^{*(1)}} (r_{i,u}^{(1)} - \bar{r}_i^{(1)})^2}{n_i^{*(1)} - 1}}$ 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 u th 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 incentives' estimates, HAIS determines the set of incentives to be applied in the Hoeffding step, \mathcal{A} (Line 8). The incentives that belong to \mathcal{A} are referred to as *active incentives*. The other incentives 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. 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.

4.2.4 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.

After the sampling step, although the algorithm now has initial estimates of the incentives' density, it might not be confident enough (i.e., the confidence level is less than

Algorithm 3 Detailed Pseudo code of 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:

$\epsilon_1, \epsilon_2, U_1, L_e, L_h, L_s, N_s$ ▷ see Table 4.2 for their descriptions

Output:

r, t , ▷ overall utility, number of periods used
 $\mathbf{N} = \{n_i^{(t)} \mid t = 1, \dots, T; i = 1, \dots, I\}$ ▷ 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:    $n_i^{(t)} \leftarrow \lfloor u_1/g_i \rfloor; r_i^{(t)} \leftarrow \text{ApplyIncentive}(i, n_i^{(t)});$ 
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 3.2;
07:   Update  $d_i^{lower}$  and  $d_i^{upper}$  using Eq. 4.3; ▷ conf. interval of incentive  $i$ 's estimate
08:  $\mathcal{A} \leftarrow \{i \mid i \in \{1, \dots, I\}, \forall j \neq i : d_j^{lower} \leq d_i^{upper}\};$  ▷ set of active incentives

Hoeffding
{
09: if  $(b_1 \geq \sum_{i \in \mathcal{A}} c_i)$  and  $(|\mathcal{A}| > 1)$  then
10:   Calculate  $U_2$  using Equation 4.16;
11:    $u_2 \leftarrow \min \left\{ U_2, \frac{b_1}{\sum_{i \in \mathcal{A}} c_i/g_i} + u_1 \right\};$ 
12:   if  $u_2 > u_1$  then ▷ check if further exploration is needed
13:      $t \leftarrow 2;$  ▷ start period 2
14:     for  $i \in \mathcal{A}$  do ▷ pull the active incentives
15:        $n_i^{(t)} \leftarrow \lfloor (u_2 - n_i^{(1)} * g_i)/g_i \rfloor; r_i^{(t)} \leftarrow \text{ApplyIncentive}(i, n_i^{(t)});$ 
16:        $b \leftarrow b - c_i n_i^{(t)}; b_1 \leftarrow b_1 - c_i n_i^{(t)};$  Update  $d_i$  using Equation 3.2;
17:  $\mathbf{A} \leftarrow \{1, \dots, I\};$  ▷ all incentives are now active

Stepped Expl.
{
18: Calculate confidence level  $l^{(t)}$  using Equation 4.17;
19:  $b_2 \leftarrow \epsilon_2 b;$  ▷ residual budget for stepped exploitation
20: 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
21:    $t \leftarrow t + 1; i \leftarrow \text{argmax}_{i \in \{1, \dots, I\} | c_i \leq b} \{d_i\};$ 
22:    $n_i^{(t)} \leftarrow \max\{1, \lfloor b_2/(c_i T_s) \rfloor\}; r_i^{(t)} \leftarrow \text{ApplyIncentive}(i, n_i^{(t)});$ 
23:    $b \leftarrow b - c_i n_i^{(t)}; b_2 \leftarrow b_2 - c_i n_i^{(t)};$  Update  $d_i$  using Equation 3.2;
24:   Calculate confidence level  $l^{(t)}$  using Eq. 4.17; ▷ to consider stop stepped exploiting

Pure Expl.
{
25:  $t \leftarrow t + 1;$ 
26: while  $b \geq \min_{i \in \{1, \dots, I\}} c_i$  do
27:    $i \leftarrow \text{argmax}_{i \in \{1, \dots, I\} | c_i \leq b} \{d_i\};$ 
28:    $n_i^{(t)} \leftarrow \lfloor b/c_i \rfloor; r_i^{(t)} \leftarrow \text{ApplyIncentive}(i, n_i^{(t)}); b \leftarrow b - c_i n_i^{(t)};$ 
29:  $r \leftarrow \sum_{t=1}^T \sum_{i=1}^I r_i^{(t)};$ 
30: return  $r, t, \mathbf{N};$ 

```

L_h) that the best incentive is the real best incentive, as it may not have enough samples of users in the estimation. Therefore, in order to obtain a higher confidence (i.e., greater than or equal to L_h) so that it can exploit effectively in the remaining periods, it applies Hoeffding's inequality to determine the minimum number of users needed for each active incentive at the end of this step. By applying this inequality, we do not have to explicitly calculate L_h after the sampling step. Specifically, the inequality helps identify the number of sampled users to obtain the confidence level of L_h . Then, if this number is less than or equal to u_1 , it means the level of confidence after the sampling step is already greater than or equal to L_h . Thus, it does not need to explore more. This means, the Hoeffding step will be skipped. However, if this number is greater than u_1 , it means the confidence level after the sampling step is less than L_h . Hence, it needs to sample more users.

In more detail, let $X_{i,1}^{(t)}, \dots, X_{i,u_i}^{(t)}$ 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 denote 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:

$$P(\bar{X}_i^{(t)} - \delta_i \geq \gamma) \leq \exp\left(\frac{-2u_i^{(t)}\gamma^2}{\beta_i^2}\right) \quad \text{and} \quad (4.4)$$

$$P(\bar{X}_i^{(t)} - \delta_i \leq -\gamma) \leq \exp\left(\frac{-2u_i^{(t)}\gamma^2}{\beta_i^2}\right), \quad (4.5)$$

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 Equations 4.4 and 4.5, we have:

$$P(\bar{X}_i^{(t)} - \delta_i < \gamma) \geq 1 - \exp\left(\frac{-2u_i^{(t)}\gamma^2}{\beta_i^2}\right) \quad \text{and} \quad (4.6)$$

$$P(\bar{X}_i^{(t)} - \delta_i > -\gamma) \geq 1 - \exp\left(\frac{-2u_i^{(t)}\gamma^2}{\beta_i^2}\right). \quad (4.7)$$

Applying Equation 4.6 to the worst (active) incentive i_1 ², 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:

²To 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 $i_1 \stackrel{\text{def}}{=} i_1^{(t)} = \operatorname{argmin}_{i \in \mathcal{A}: c_i \leq b^{(t)}} \{d_i^{(t)}\}$ and $i_2 \stackrel{\text{def}}{=} i_2^{(t)} = \operatorname{argmax}_{i \in \mathcal{A}: c_i \leq b^{(t)}} \{d_i^{(t)}\}$, where $b^{(t)}$ is the residual budget after finishing period t .

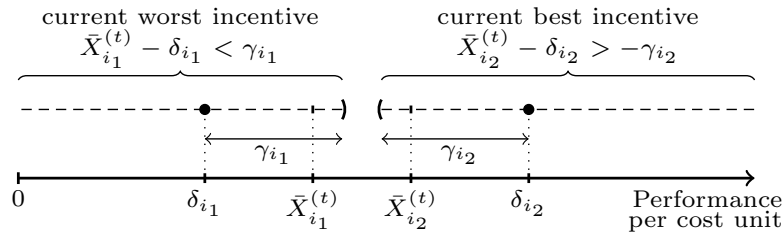


FIGURE 4.3: Illustration for Equation 4.10.

$$\gamma_{i_1} = \beta_{i_1} \sqrt{\frac{\ln(1/(1-l_1^{(t)}))}{2u_{i_1}^{(t)}}}. \quad (4.8)$$

Similarly, applying Equation 4.7 to the best incentive i_2^2 , the resulting confidence level that $\bar{X}_{i_2}^{(t)} - \delta_{i_2} > -\gamma_{i_2}$ is $l_2^{(t)} = 1 - \exp(-2u_{i_2}^{(t)}\gamma_{i_2}^2/\beta_{i_2}^2)$, or:

$$\gamma_{i_2} = \beta_{i_2} \sqrt{\frac{1}{2u_{i_2}^{(t)}} \ln(1/(1-l_2^{(t)}))}. \quad (4.9)$$

To differentiate the effectiveness of the two incentives, the margins of error γ_{i_1} and γ_{i_2} must be small enough compared to the distance between the expected values of the two incentives' densities:

$$\gamma_{i_1} + \gamma_{i_2} \leq \delta_{i_2} - \delta_{i_1}. \quad (4.10)$$

This is illustrated in Figure 4.3. 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 Equations 4.8, 4.9, and 4.10, we have:

$$\beta_{i_1} \sqrt{\frac{\ln(1/(1-l_1^{(t)}))}{2u_{i_1}^{(t)}}} + \beta_{i_2} \sqrt{\frac{\ln(1/(1-l_2^{(t)}))}{2u_{i_2}^{(t)}}} \leq \delta_{i_2} - \delta_{i_1}. \quad (4.11)$$

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 thesis. 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 Equations 4.8 and 4.9, 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 Equation 4.11 we have:

$$u^{(2)} \geq \frac{\ln(1/(1 - \sqrt{L_h})) (\beta_{i_1} + \beta_{i_2})^2}{2(\delta_{i_2} - \delta_{i_1})^2} \stackrel{\text{def}}{=} U_2. \quad (4.12)$$

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

$$\delta_i \approx d_i^{(1)}, \quad (4.13)$$

$$\beta_{i_1} \approx b_{i_1}^{(1)} \stackrel{\text{def}}{=} \max_{1 \leq u \leq u_{i_1}^{(1)}} \{X_{i_1, u}^{(1)}\} - \min_{1 \leq u \leq u_{i_1}^{(1)}} \{X_{i_1, u}^{(1)}\}, \text{ and} \quad (4.14)$$

$$\beta_{i_2} \approx b_{i_2}^{(1)} \stackrel{\text{def}}{=} \max_{1 \leq u \leq u_{i_2}^{(1)}} \{X_{i_2, u}^{(1)}\} - \min_{1 \leq u \leq u_{i_2}^{(1)}} \{X_{i_2, u}^{(1)}\}. \quad (4.15)$$

Therefore, from Equation 4.12 we have:

$$U_2 \approx \frac{\ln(1/(1 - \sqrt{L_h})) (b_{i_1}^{(1)} + b_{i_2}^{(1)})^2}{2(d_{i_2}^{(1)} - d_{i_1}^{(1)})^2}. \quad (4.16)$$

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 11) by approximating the maximum number of users based on the total cost of applying the active incentives ($\sum_{i \in \mathcal{A}} c_i/g_i$). 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 14–16).

4.2.5 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 L_s , e.g., 90%. The confidence level in finding the real best incentive at the end of period t can be calculated from Equation 4.11. 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 ($\forall i = 1, \dots, I$), β_{i_1} , and β_{i_2} with the estimates so far: $\delta_i \approx d_i^{(t)}$, $\beta_{i_1} \approx b_{i_1}^{(t)}$, and $\beta_{i_2} \approx b_{i_2}^{(t)}$. Thus, from Equation 4.11, we have the maximum confidence level in finding the real best incentive at the end of period t :





























$$l^{(t)} \approx \left(1 - \exp \left(- \frac{2(d_{i_2}^{(t)} - d_{i_1}^{(t)})^2}{\left(b_{i_1}^{(t)} / \sqrt{u_{i_1}^{(t)}} + b_{i_2}^{(t)} / \sqrt{u_{i_2}^{(t)}} \right)^2} \right) \right)^2. \quad (4.17)$$

Equation 4.17 is used before conducting stepped exploitation (Line 18) and at the end of each loop in this step (Line 24) to decide whether to continue stepped exploiting or not (the third condition in Line 20).

Additional information can also be used together with the condition about L_s to consider stopping stepped exploiting sooner. Specifically, we use the number of consecutive periods that the current best incentive has been applied. If in this step, an incentive has been applied consecutively in the last 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 20) 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 .

4.2.6 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 Tran-Thanh et al. (2010), as it is simple and efficient. It is referred to as *pure exploiting* in this thesis. 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.

| DAY | 1 | 2 | 3 | 4 | 5 | |
|--------|--|--|--|--|--|--|
| GROUPS |       |       |   |     |     |       |
| COST | £32 | £8 | £8 | £8 | £24 | |
| STEP | Exploration | Stepped Exploitation | | | Pure Exploitation | |
| PHASE | | Exploitation | | | | |


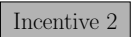
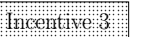




FIGURE 4.4: An example of applying policy with **Stepped ϵ -first**, where $I = 3$, $T = 5$ (periods), $B = \text{£}80$, $g_1 = 2$, $g_2 = 4$, $g_3 = 2$, $c_1 = \text{£}2$, $c_2 = \text{£}4$, $c_3 = \text{£}2$, $\epsilon_1 = 0.40$, $\epsilon_2 = 0.50$, incentive 1 is the real best incentive, and incentive 3 is the real worst incentive.

4.3 Experimental Evaluation

To systematically evaluate the performance of HAIS, we use 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, 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 (Subsection 4.3.1), the experimental settings (Subsection 4.3.2), the corresponding results (Subsection 4.3.3), and the practical usage of the HAIS algorithm based on the results (Subsection 4.3.4). This section corresponds to Contribution 3 as presented in Section 1.3.

4.3.1 Benchmarks

As state-of-the-art algorithms are not specifically designed to deal with the time constraint of the ISP, we make a number of minor modifications:

(1) **BOIS** . The algorithm will be described in detail in Section 5.2. When applying BOIS into the ISP1, it considers an incentive as a cluster which has only one incentive. Although both HAIS and BOIS are to deal with the ISP, they are to solve different variants of the problem. HAIS is for the ISP1 where the incentives are uncorrelated, while BOIS is for the ISP2 where the incentives are correlated. Hence, we run BOIS as a benchmark for HAIS is to evaluate the performance of BOIS on the ISP1 in comparison with HAIS.

(2) **ϵ -first.** This algorithm 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 (Section 2.2.2.1). Then, it spends the subsequent period purely exploiting the best incentives with the residual budget, i.e., $(1 - \epsilon_1)B$ as mentioned in Subsection 4.2.6. 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.

(3) **Stepped ϵ -first.** 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 exploring phase, the best incentive might not be the real best incentive and this may only be discovered by a number of times the best incentive is applied while exploiting. Thus, we apply the stepped exploitation of **H AIS** 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 **H AIS**, it spends the last period purely exploiting.

To illustrate, Figure 4.4 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 Figure 4.2. In the exploration phase (day 1) of this example, as it does not look at the group sizes (as per **H AIS**), it applies the incentives evenly (4 times each). After this period, suppose that the estimate of incentive 2 is higher than those of the other incentives. So, **Stepped ϵ -first** identifies that incentive 2 is the best one. Compared with **ϵ -first**, it is better as instead of applying incentive 2 (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 2) and updates this incentive's estimate. In so doing, it identifies that incentive 2 is not the best any more. Hence, on day 3, it applies incentive 1 (the new best incentive). We also suppose that the estimates after days 3 and 4 are consistent with incentive 1 being the best, thus it simply applies this incentive in days 4 and 5. Compared with the example of **H AIS** in Figure 4.2, as **H AIS** 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 1) 6 times compared to **Stepped ϵ -first**, which only applies it 4 times. Hence, it better estimates this incentive and finds that this is the best incentive. In the exploitation phase, **H AIS** applies this incentive all the time. However, with **Stepped ϵ -first**, after the exploration phase, it applies incentive 2 twice (in period 2) before identifying and applying the real best incentive (incentive 1) 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.

(4) **Stepped fKUBE**. This algorithm (Section 2.2.2.1) applies all the incentives once to obtain initial estimates of the densities of the incentives. This can be considered as an initial exploration step. Then, it applies stepped and pure exploiting techniques as per HAIS. The only difference is that **Stepped fKUBE** uses the upper confidence bounds (UCBs) of the densities' estimates instead of the densities that HAIS uses. In more detail, 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's density. The UCB of the estimate of incentive i is:

$$d_i^{*(t)} = \frac{\hat{\mu}_i^{(t)}}{c_i} + \frac{r_{\min} + (r_{\max} - r_{\min})\sqrt{(2 \ln u^{(t)})/u_i^{(t)}}}{c_i}. \quad (4.18)$$

In Equation 4.18, $u^{(t)} = \sum_{i=1}^I u_i^{(t)}$ is the total number of users in all incentives until the end of period t and r_{\min} (r_{\max}) is the minimum (maximum) value of the utility, which is identified in Table 4.1. We will discuss this table in Section 4.3.2.1. Discussion about the meaning of Equation 4.18 can be found in Subsection 2.2.2. Finally, in the last period, fKUBE purely exploits.

(5) **Survival of the Above Average (SOAAv)**. This algorithm (Section 2.2.2.1) applies different incentives from round to round. In each round, it applies the incentives that have estimates above $(1 + x)$ times the average of incentives' estimates in the previous period once. 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 make use of the residual budget.

(6) **Exp3**. This algorithm (Auer et al., 2002b) maintains a weight list where each item corresponds to an incentive. The weights will be 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 of in the other periods before the deadline, which is shown in Equation 4.20. In period $t = 2, \dots, T - 1$, the probability of choosing incentive i is:

$$p_i^{(t)} = (1 - \gamma) \frac{w_i^{(t)}}{\sum_{j=1}^I w_j^{(t)}} + \frac{\gamma}{I}. \quad (4.19)$$

Then, the received utility, $r_i^{(t)}$, will be used to update the weights of the incentives to prepare for the next period:

$$w_j^{(t+1)} = \begin{cases} w_j^{(t)} \exp\left(\frac{\gamma}{I c_j p_j^{(t)}} \cdot \frac{r_j^{(t)} - r_{\min}}{r_{\max} - r_{\min}}\right), & \text{if } j = i \\ w_j^{(t)}, & \text{otherwise} \end{cases}. \quad (4.20)$$

In the last period, **Exp3** conducts pure exploration as in **H AIS**.

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

4.3.2 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 only 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 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., $B, T, I, g_i, c_i, \mu_i, \sigma_i, \delta_i \forall i = 1, \dots, I$ (except the corresponding independent variable) are generated randomly in specific ranges. The ranges of the quantities are shown in Table 4.1 and will be discussed in more detail later in Subsection 4.3.2.1. 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 randomly 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 **H AIS**) 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 **H AIS**. 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. Error bars of the line graphs in Figures 4.5–4.17 represent the confidence intervals. Regarding the simulation to better understand the algorithms' behaviours, we run with six incentives where the densities of incentives 1 to 5 are 90, 80, 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 $0.4\mu_i \forall i = 1 \dots 6$ (the mean value of the range presented in Table 4.1 which will be discussed in the next subsection). The group size of each incentive in each period is generated randomly in the range from 1 to 10. We also run the simulation 20,000 times as with the above-mentioned simulations.

Next, in Subsection 4.3.2.1, we detail the ranges of the quantities used for randomisation in the simulations. Then, in Subsection 4.3.2.2, we detail the values of the predefined parameters of the algorithms. Finally, in Subsection 4.3.2.3, we present how the performance of a group is generated in the simulations (based on the performance of the individuals of the group).

4.3.2.1 Ranges of the Quantities for Randomisation

The ranges of the quantities are described in Table 4.1. The values are chosen to represent realistic settings from a number of real crowdsourcing projects. 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 Subsection 4.3.3, 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

TABLE 4.1: 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 per \mathcal{L} |
| 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 |

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 [Yang et al. \(2008\)](#), 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 ([Ramchurn et al., 2013](#)). 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 high chance to win the competition ([Yang et al., 2008](#)). 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 4.1), 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. 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 about U_1 sampled users. According to our calculation, the budgets used for the first crowdsourcing project in [Mason and Watts \(2010\)](#) (experiment 1: image ordering) and the crowdsourcing project in [Yin and Chen \(2015\)](#) (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 *HAI*S, *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 mean value of the range of the group sizes as described in [Table 4.1](#), 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 likely 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.

TABLE 4.2: 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). |
| | 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. |
| BOIS | ϵ_1 | 0.10 | Budget limit for exploration. |
| | ϵ_2 | 0.50 | Budget for stepped exploitation (calculated based on the residual budget). |
| | U_1 | 20 | Target number of sampled users to obtain after the first (sampling) period. |
| ϵ -first | ϵ_1 | 0.10 | Budget limit for exploration. |
| Stepped ϵ -first | ϵ_1 | 0.10 | Budget limit for exploration. |
| | 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 | x | 0 | $x = 0$ 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. |

4.3.2.2 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.

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 4.3.3, we only present the results on the simulations with the values of the algorithms' predefined parameters as described in Table 4.2. Regarding the predefined parameters of

HAIS, as most of them are self-explanatory and some of them are already discussed in Subsection 4.2.2, we do not explain them here.

4.3.2.3 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 users there are in a group, the better the performance of the whole group. Specifically, for every incentive $i = 1, \dots, I$, suppose $\mu_i^{(u)}$ is the mean utility of a user in the group (corresponding to this incentive). Then, the mean utility of the whole group is $\mu_i = g_i \mu_i^{(u)}$, where g_i is the group size of incentive i . 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 Araujo (2013). 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.

4.3.3 Results

In general, HAIS performs the best in most cases (Figures 4.5–4.11). In more detail, HAIS performs better with a larger financial budget (Figure 4.5), with a looser deadline (Figure 4.6), with fewer incentives (Figure 4.7), and with smaller values of the standard deviation of the incentives' utilities (Figure 4.8). Its performance is reasonably stable with different group sizes (Figure 4.9), 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 (Figure 4.10). Additionally, as shown in Figure 4.9, 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 better when the group size of the best incentive is the largest (Figure 4.10) than when the group size of the worst incentive is the largest (Figure 4.11). In both settings, HAIS is shown to remain almost 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.

Regarding BOIS, the purpose of this benchmark is to see how an algorithm designed for the ISP2 to be applied into the ISP1 compared to HAIS. With this purpose, we focus more on the comparison between HAIS and BOIS than the other algorithms. So, we discuss

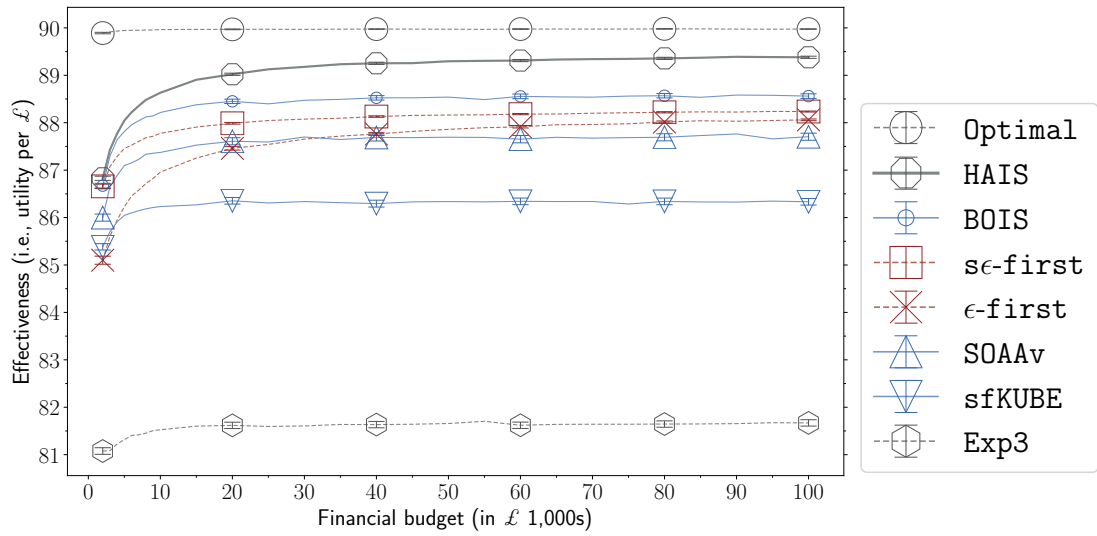


FIGURE 4.5: Performance of Algorithms for Different Financial Budget Sizes

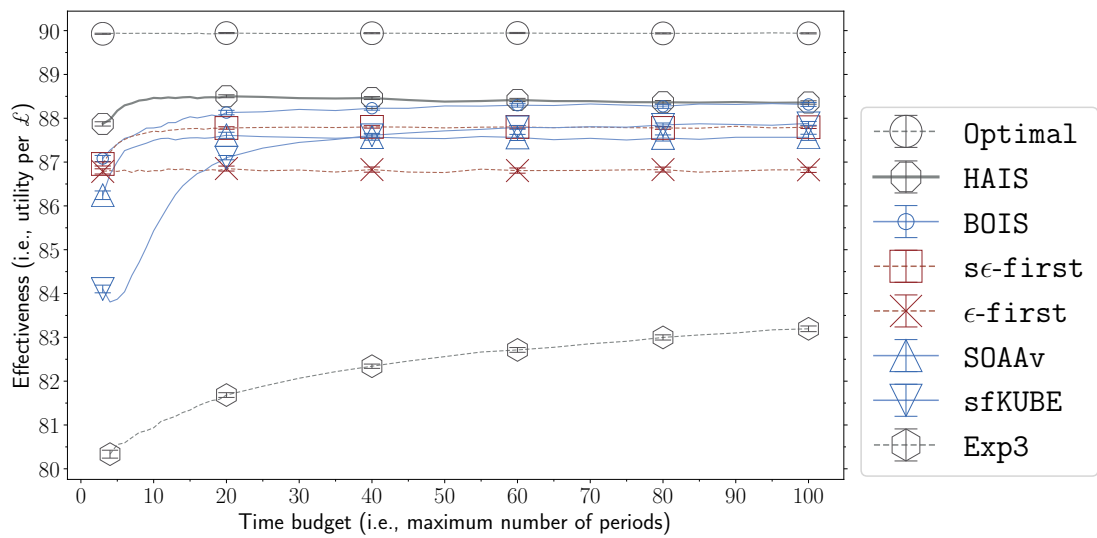


FIGURE 4.6: Performance of Algorithms for Different Time Budget Values

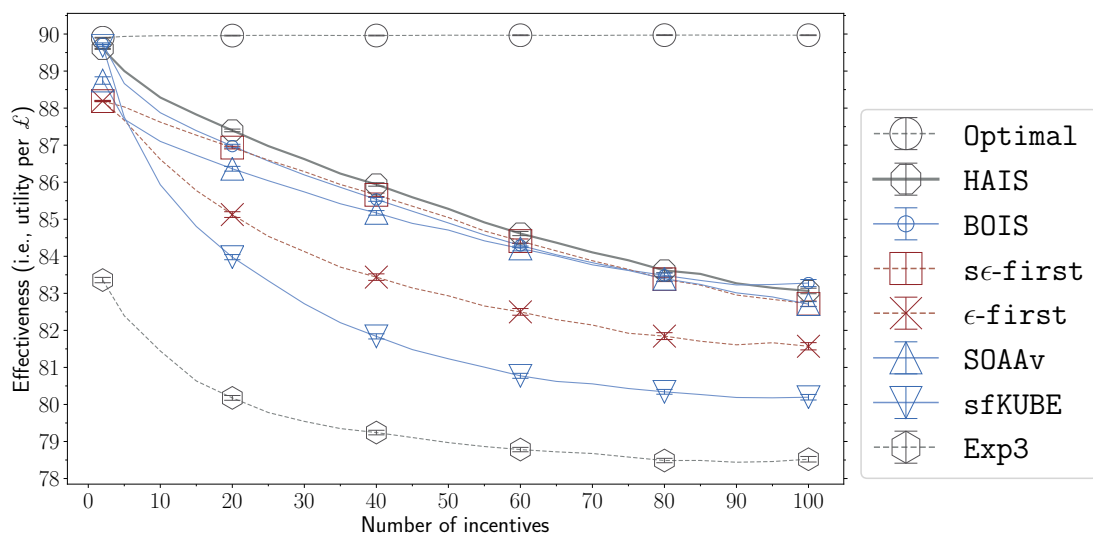


FIGURE 4.7: Performance of Algorithms for Different Numbers of Incentives

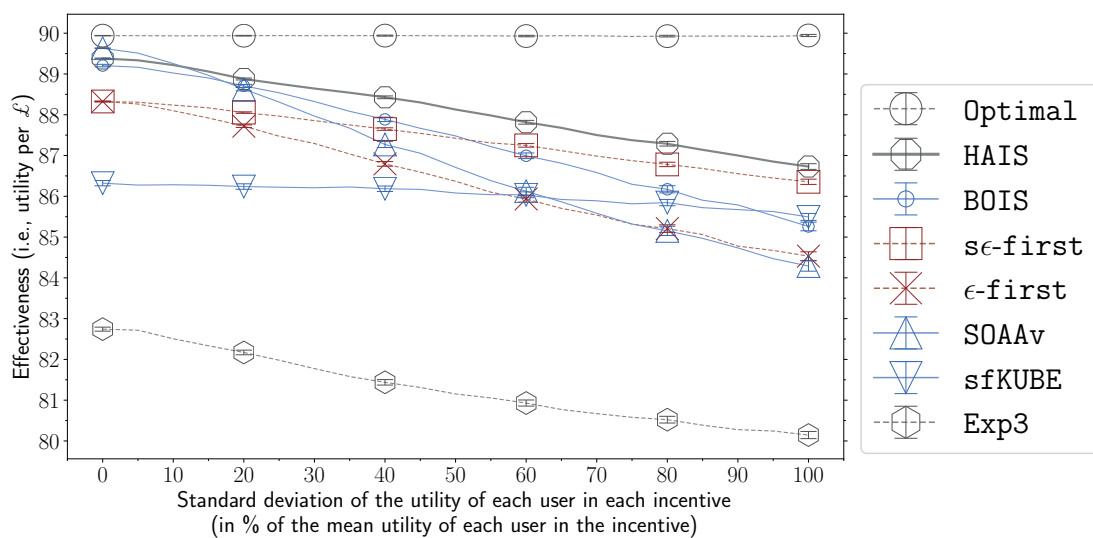


FIGURE 4.8: Performance of Algorithms for Different Values of the Standard Deviation of the Incentives' Utilities

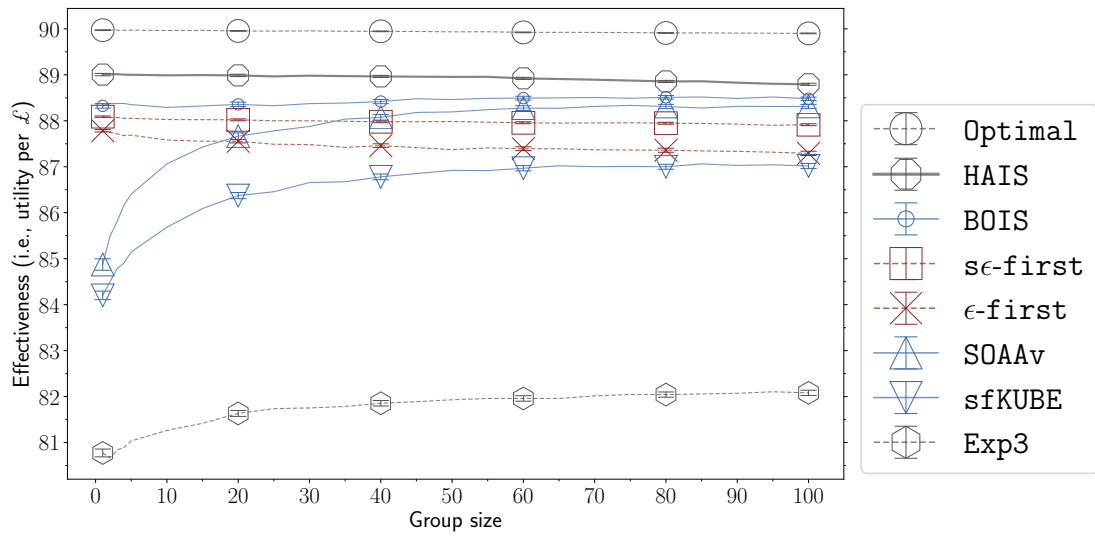
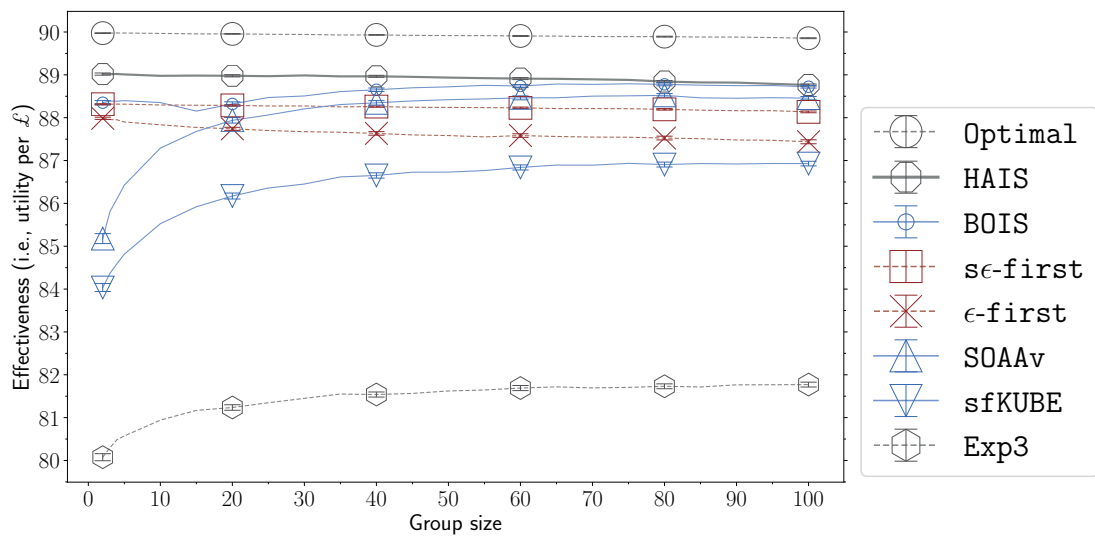


FIGURE 4.9: Performance of Algorithms for Different Values of Maximum Group Size

FIGURE 4.10: 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 $x - 1$.

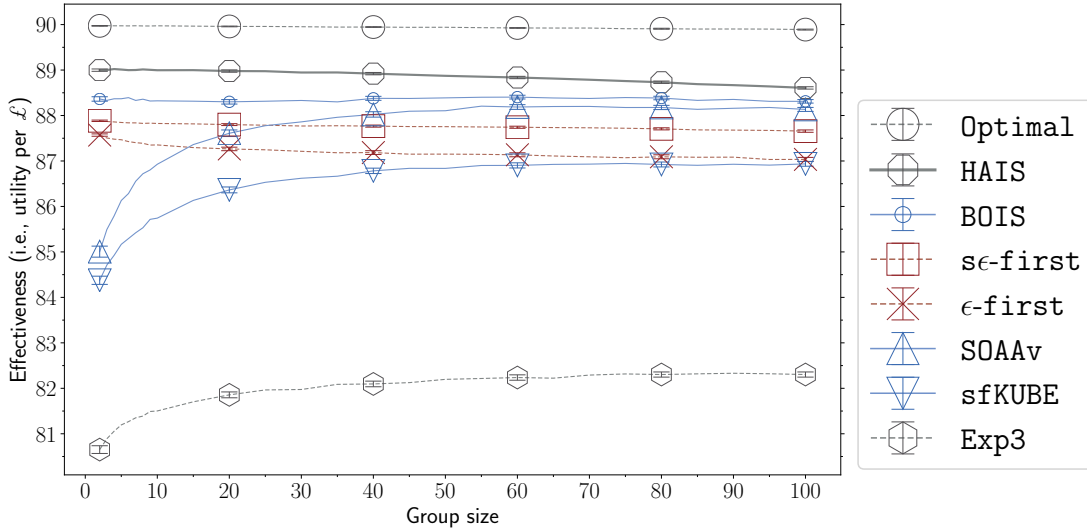


FIGURE 4.11: Performance of Algorithms for Different Values of Maximum Group Size on the Worst Incentive. Group size of the worst incentive is fixed with the value of the independent variable, x , while group sizes of the other incentives are generated randomly from 1 to $x - 1$.

the performance of BOIS before other algorithms. In general, BOIS is worse than HAIS in most cases and is better than the other algorithms in many cases. The main difference between HAIS and BOIS is shown in Figure 4.8. In more detail, when the standard deviation of the utility of each incentive becomes large, the performance of BOIS drops more clearly compared to that of HAIS. This is because HAIS takes advantage of the uncorrelated relationship between the incentives to simultaneously (and hence quickly) explore highly effective incentives in the Hoeffding step. As designed to deal with the correlated ISP, BOIS cannot do similarly since spending the budget on this might not be effective when the number of incentives is large. However, as shown in Figure 4.6, the performance of BOIS approaches that of HAIS when the time budget becomes large. This is because the stepped exploitation of BOIS can reduce the consequence of the above weakness (that does not quickly explore the incentives) by making use of the large time budget to explore (while exploiting) the best incentives. These results say that it is better to use HAIS instead of BOIS to deal with the ISP1.

Regarding Exp3, as in general this algorithm 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 (Figures 4.5–4.11). This is because choosing the incentives randomly based on their weights does not work well when the time budget is small. As reflected in Figure 4.6, 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). Regarding 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 Subsection 4.3.3.2 when explaining the importance and the usage of stepped exploitation.

Next, we will discuss each of the above-mentioned reasons of why H AIS performs effectively in the following subsections. In more detail, we will present the exploration-exploitation strategy (Subsection 4.3.3.1), the way it makes use of the time budget (Subsection 4.3.3.2), and the elimination technique (Subsection 4.3.3.3). Then, we will continue with effective ways to use H AIS in a specific crowdsourcing project (Subsection 4.3.4).

4.3.3.1 Exploration-exploitation Balance

Regarding the exploration-exploitation strategy, as both financial and time budgets are limited in the ISP1, 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, the performance of **Stepped fKUBE** 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 Figures 4.6 and 4.9, the performance of this algorithm improves significantly when the time budget or the group sizes become large. This is due to the more periods, the more time for the algorithm to identify the best incentive. Also, with larger group sizes, it has more sampled users, and hence the initial estimates become better. Whereas, in most cases, **Stepped ϵ -first** performs better than **Stepped fKUBE** (Figures 4.5–4.11). The reason is that, **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. Yet, the performance of **Stepped ϵ -first** depends on choosing an appropriate value of ϵ_1 . On the other hand, as it is using Hoeffding’s inequality, H AIS is more flexible in determining an appropriate budget for exploration. Actually, Figure 4.12 shows that when the budget for the crowdsourcing project (B) is large, instead of using all $\epsilon_1 B$ as in **Stepped ϵ -first**, H AIS 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 Subsection 4.3.3.3.

4.3.3.2 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 thus improves

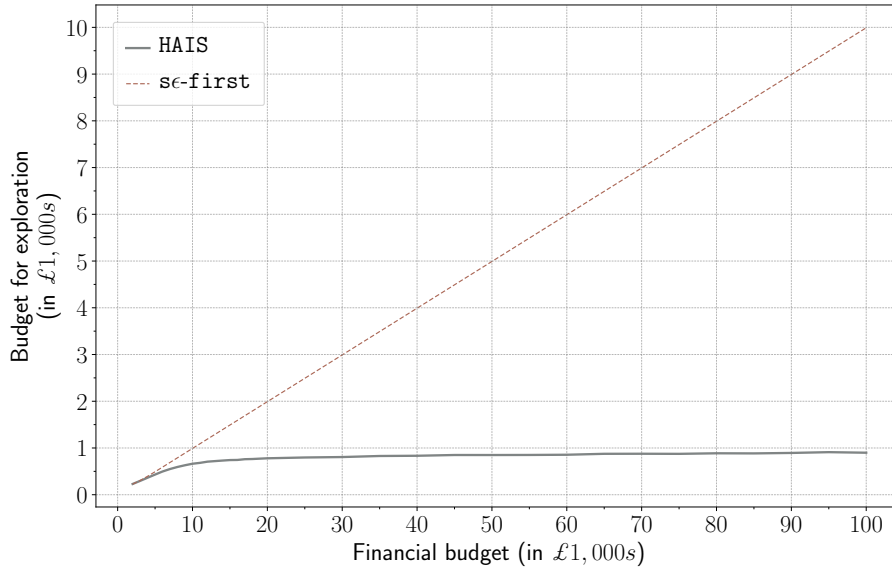
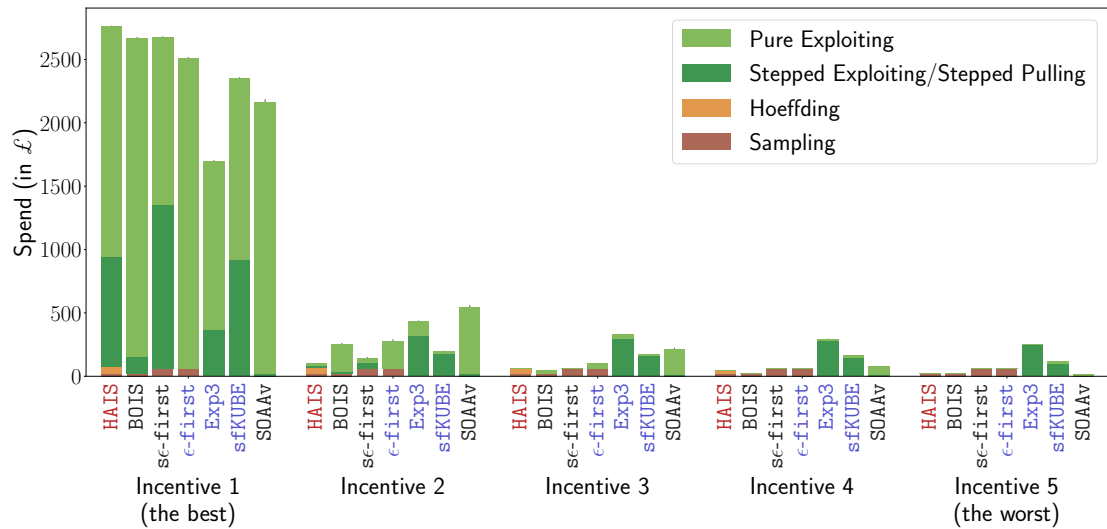


FIGURE 4.12: Average Budget Used for Exploration. This is the corresponding result of the simulations shown in Figure 4.5. 99% confidence intervals are omitted as they are too small.

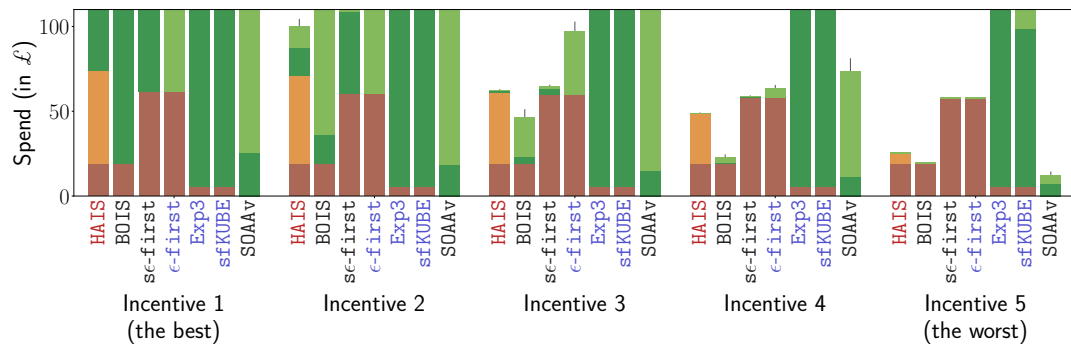
the overall performance of the algorithm significantly. Actually, Figures 4.5–4.11 show that **Stepped ϵ -first** performs significantly better than ϵ -first, especially when the budget is not large or when the time budget is large enough. Specifically, in Figure 4.5, with low budgets (e.g., from £500 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 Figure 4.6, this difference in performance between the two algorithms is clearer when the time budget is large enough (e.g., larger 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** (Figures 4.5–4.11). This is especially the case when the financial budget is large (Figure 4.5). Actually, 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). Hence, 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 Figure 4.5 where ϵ -first’s effectiveness approaches that of **Stepped ϵ -first** when the budget size becomes larger.

In addition, as discussed in Section 4.2.5, **HAIS** is able to stop stepped exploiting sooner without significantly affecting the results. Hence, setting a loose deadline is better



(a) The whole figure



(b) A zoom of (a)

FIGURE 4.13: Cost Distribution Over the Incentives Across the Phases of Each Algorithm. ϵ -first is Stepped ϵ -first and sfKUBE is Stepped fKUBE.

for its performance as it has enough time to conduct stepped exploitation effectively. Actually, Figure 4.17 shows the average number of periods used by HAIS in the setting corresponding to Figure 4.6. 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.

4.3.3.3 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 eliminating 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 Figures 4.13, 4.14, and 4.15. In more detail, Figure 4.13a shows that, compared with other algorithms, HAIS spends more of its budget on the best incentive (incentive 1) and less of its budget on the other incentives. By looking more closely at how the cost is distributed over the incentives across the phases of HAIS (Figure 4.13b), we can see that after the sampling step, HAIS identifies ineffective incentives effectively. This figure 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. This helps HAIS not only partially exploit highly effective incentives while exploring but also increase 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 (Figures 4.14a and 4.15). Additionally, Figure 4.14b 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 result in a good performance. Actually, Figure 4.14 shows that SOAAv under-explores the incentives, especially the highly effective ones in the first (e.g., 3) periods. This results in exploiting less effective incentives in the rest periods. More specifically, right 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 can be eliminated with a probability that is not insignificant. Therefore, in the last periods (e.g., from period 4 to period 9), SOAAv tends to apply the ineffective incentives much more than HAIS and **Stepped ϵ -first** (Figure 4.14). One exception that SOAAv performs better than HAIS is the case when the difference in the effectiveness of the incentives is small. In detail, Figure 4.8 shows that when the standard deviation of the utility of each incentive is less than about 20% of the mean utility of the incentive, SOAAv enjoys slightly higher overall utility than HAIS. The reason is that, right after the first period, the estimate of the density 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 next periods, these incentives will be eliminated. However, as in a crowdsourcing project, the requester will try their best to determine good enough incentives for the selection, the differences in effectiveness between the incentives tend to be not too clear as in this case. Therefore, as discussed

in Section 4.3.2.1, we do not include the case in the simulations by focusing on more realistic cases where the differences are large enough.

4.3.4 Practical Usage of the HAIS Algorithm

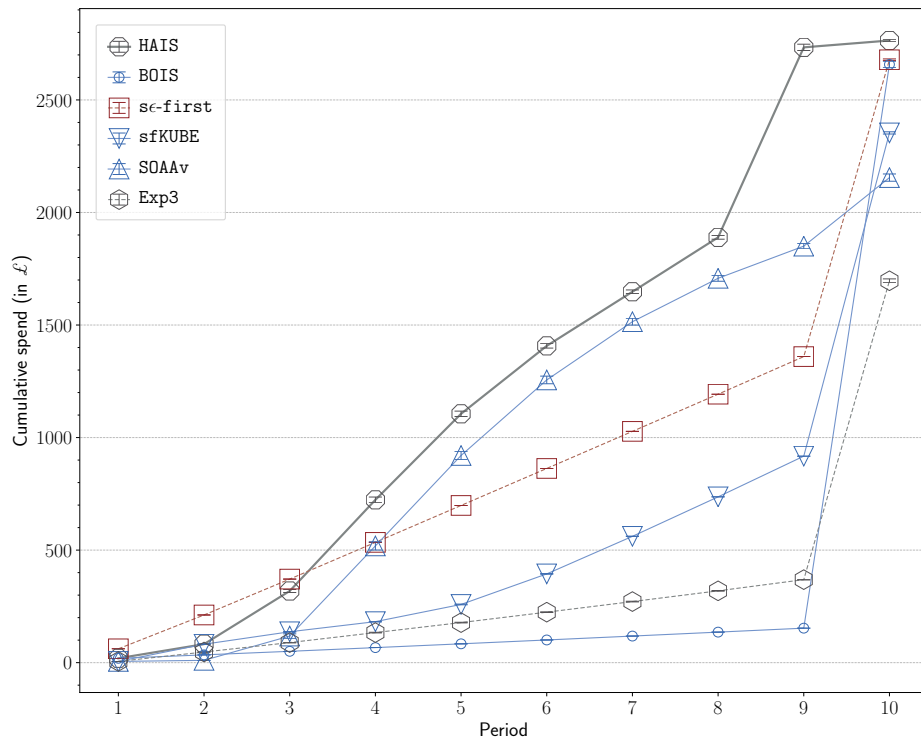
The above-mentioned results suggest several guidelines for using this algorithm effectively in practice. First, the larger the budget the better (Figure 4.5). It is reasonable that when the budget is larger, HAIS can spend more on exploring the incentives so that it can better understand before exploiting.

Second, the fewer incentives the better (Figure 4.7). Specifically, when there are more incentives, it 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 projects, they do not have good reasons to eliminate some chosen candidate incentives so as to improve the overall performance of HAIS. Moreover, it is not clear how the current number of candidate incentives affects the overall performance. Therefore, we run another simulations to investigate HAIS' performance with different ratios of the financial budget compared to the round cost. The result is shown in Figure 4.16. This figure is consistent with Figure 4.5, that the larger the budget is the better HAIS performs. It can be seen from this figure that, the performance of HAIS increases significantly when the budget ratio 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.

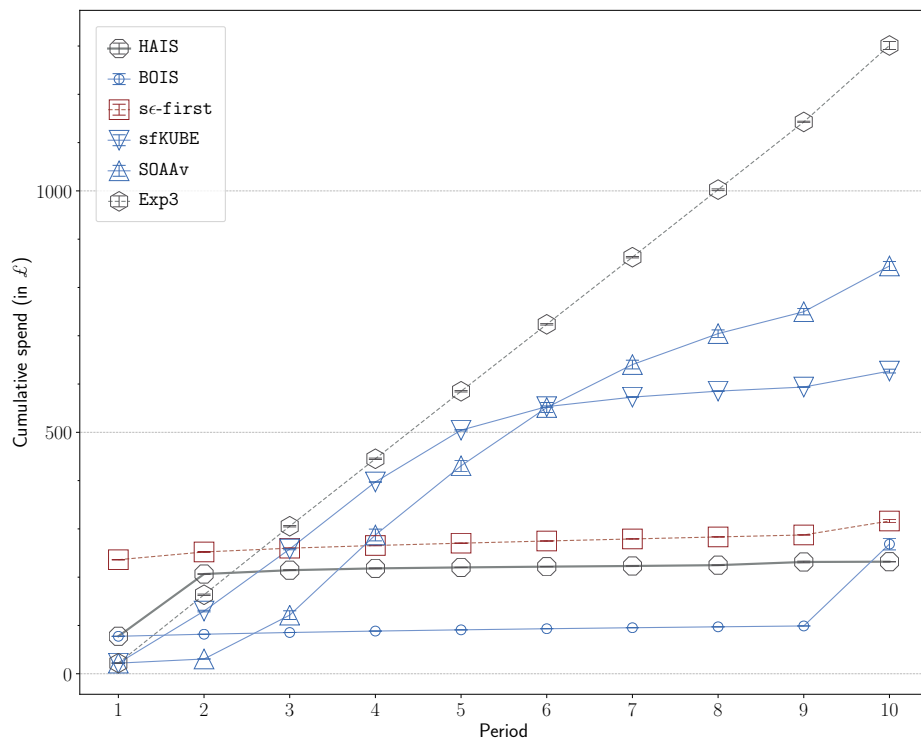
Third, the time budget should be large enough (e.g., from 15 to 20 periods), but does not need to be very large (e.g., 100 periods) so that HAIS has enough time to conduct stepped exploitation effectively (Figure 4.6). Also, as discussed in Section 4.2.5, HAIS is able to stop stepped exploiting sooner without affecting significantly the results. Hence, setting a loose deadline is better for its performance as it has enough time to conduct stepped exploitation effectively. Actually, Figure 4.17 shows the average number of periods used by HAIS in the setting corresponding to Figure 4.6. This figure shows that, although the time budget is large, HAIS tends to use a lot less. 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 long deadline (but not necessarily too long). The algorithm will then automatically select an appropriate time to stop.

4.4 Summary

In this chapter, we presented the ISP1 (or uncorrelated ISP), a variant of the ISP where the candidate incentives are not correlated. We then developed HAIS, an adaptive



(a) Total spent on the real best incentive (a zoom of this graph on the first three periods is shown in Figure 4.15)



(b) Total spent on the other incentives

FIGURE 4.14: Cost Distribution Across the Periods Incurred by Each Algorithm

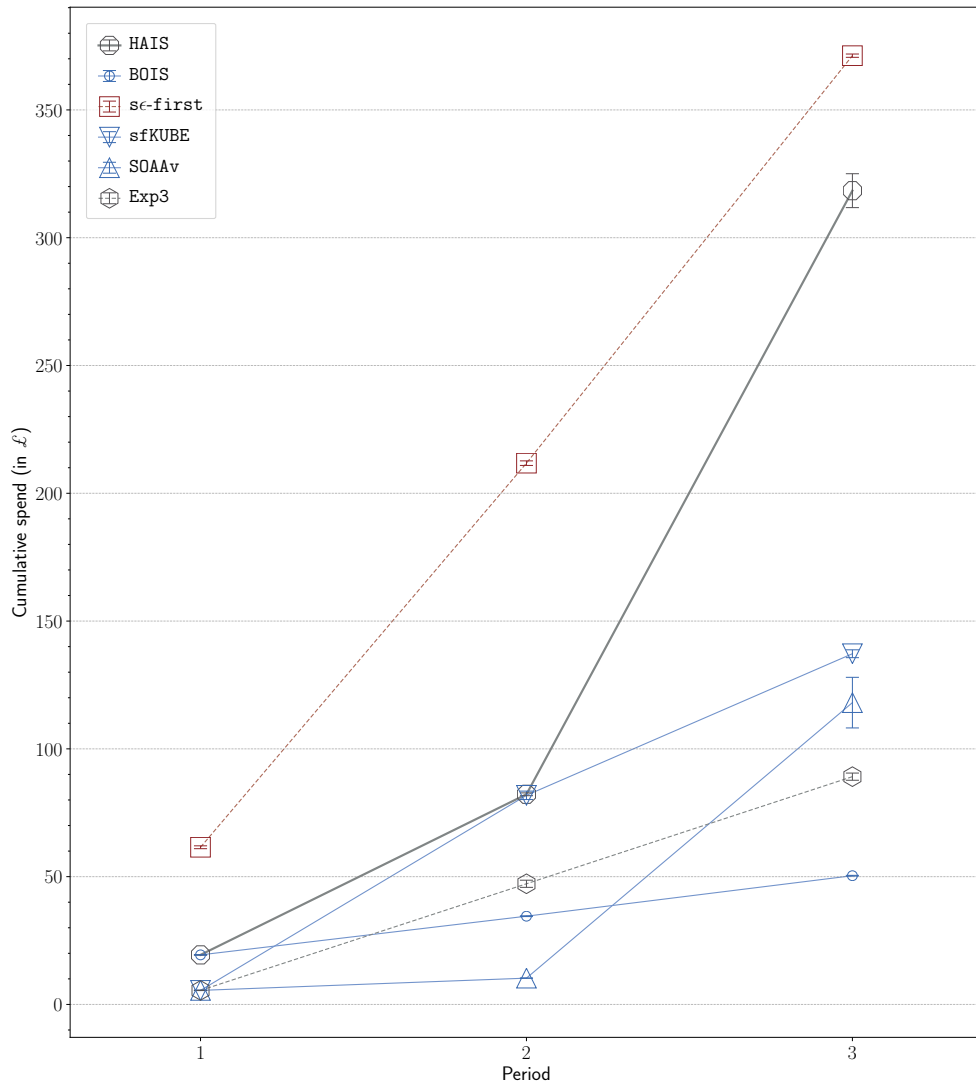


FIGURE 4.15: The first three periods of Figure 4.14a

algorithm, to solve the ISP1 efficiently. The algorithm takes into account the group-based nature of the incentives and uses Hoeffding's inequality to adaptively determine how much exploration on each incentive is sufficient. It also makes use of the time budget to conduct further exploration efficiently while exploiting. Additionally, HAIS is shown to have a good way to eliminate ineffective incentives, so that it can spend more budget on highly effective ones. The performance of the algorithm was then evaluated by simulations in a wide range of settings. Specifically, we run the simulations with different financial budget sizes, different time budget values, different numbers of incentives, different values of max group size, and different values of the standard deviation of the incentives' utilities. We also run the simulations with some other settings to have a closer look at how the algorithms perform in particular ways. The results of the simulations show that HAIS performs the best in most cases. Throughout this chapter, we obtain the first three contributions as presented in Section 1.3. Specifically, the model (i.e., the formal presentation of the ISP1 presented in Section 4.1) is Contribution 1,

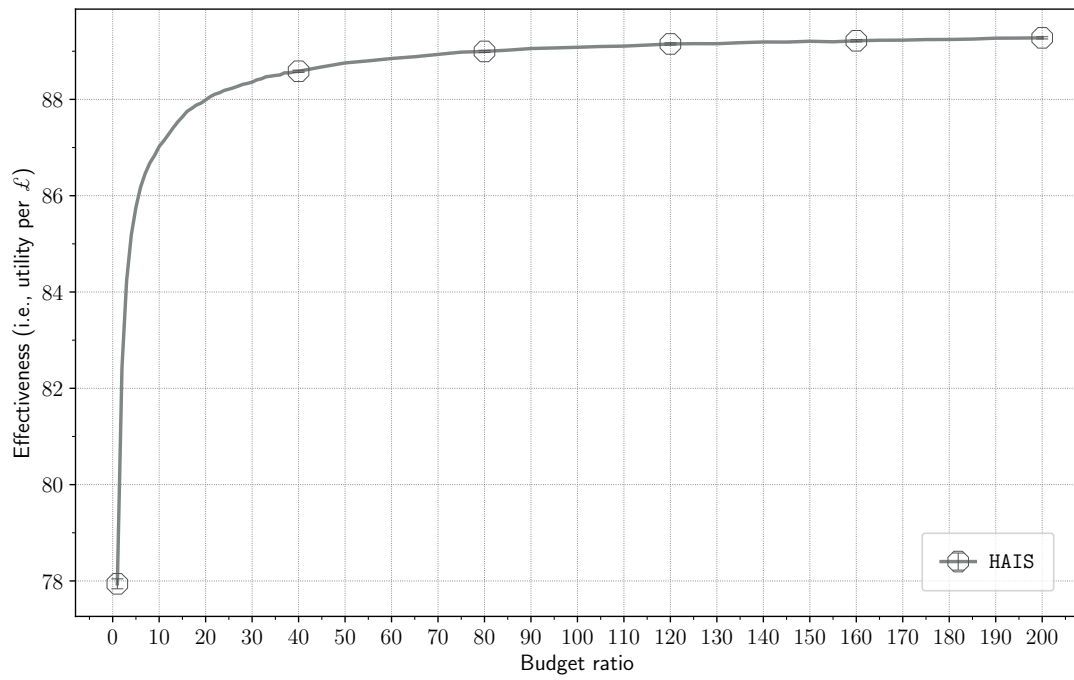


FIGURE 4.16: Performance of HAIS for Different Budget Ratios. Budget ratio is $B/\text{round_cost}$.

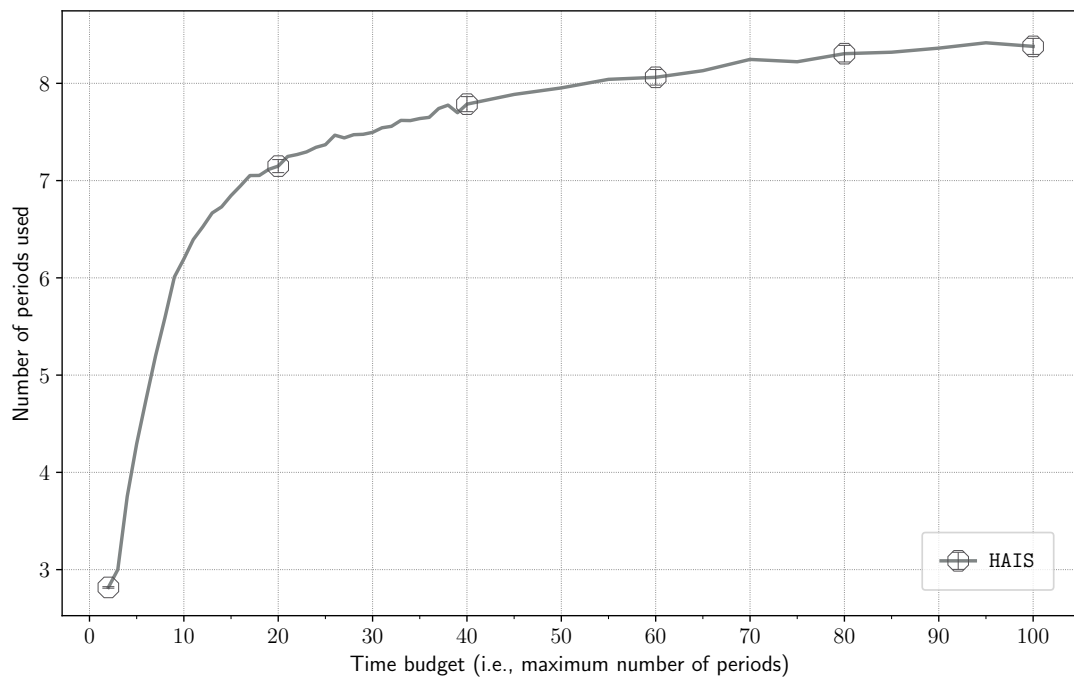


FIGURE 4.17: Number of Periods Used by HAIS

the HAIS algorithm (detailed in Section 4.2) is Contribution 2, and the evaluation of the algorithm's performance (Section 4.3) is Contribution 3.

The algorithm helps us achieve the first three objectives of the research as mentioned in Section 1.2. More specifically, in Subsection 4.3.3, HAIS is shown to be efficient (**RO1**). Also, as presented in Section 4.2, with a specific crowdsourcing project, it can be used in an autonomic manner (**RO2**) without the need to tune any predefined parameters (**RO3**). However, HAIS can only be used in crowdsourcing projects when the candidate incentives do not have any correlations. This is the case when we have prior information about user performance in the project of interest or when the budget for the project is low, so in each incentive methods, we have to continue choosing some candidate incentives. Therefore, to attain the last research objective (**RO4**), in which the solution is complete (i.e., can be applied to any crowdsourcing project), we need to solve the second variant of the ISP, the ISP2, that is when the candidate incentives are correlated. Indeed, as discussed in Subsection 3.2.3, we do not need to directly solve the third variant of the ISP, the ISP3, as we can make use of the solution to the ISP1 or the ISP2 to this variant. We will deal with the ISP2 in the following chapter.

Chapter 5

The ISP with Correlated Candidate Incentives

We are trying to solve the incentive problem (i.e., providing incentives to motivate user participation in microtask crowdsourcing) by dealing with the incentive selection problem (ISP). That is to, instead finding the best incentive in a crowdsourcing project, we choose some good candidate incentive methods (i.e., clusters) and then have a good strategy to learn the best incentive so as to maximise the overall utility of the requester. In Chapter 4, we developed an algorithm to solve the ISP1, a variant of the ISP when we have chosen some representative incentives in each chosen incentive method because we have prior knowledge about user performance in the project or the budget for the project is small compared to the chosen incentive methods. However in some projects, we have to (or we do not want to) choose such candidate incentives in each cluster. We just want to keep all of these candidates and then find a solution to effectively select the best one. This is when we have very little or no prior knowledge about the user performance in the project and the financial budget is large enough. This results in another variant of the ISP which is called the ISP2 (or correlated ISP). The ISP2 is different from the ISP1 in terms of the possible correlations between incentives in each cluster. In the ISP1, as we have chosen some representative incentives in each cluster, these correlations do not exist. So, we can only focus on the learning process, which is to identify the best incentive among the chosen ones. Yet, in the ISP2, as all the incentives in each cluster are kept, the correlations exist. Thus, to deal with this problem effectively we have to consider these correlations. That means, in the ISP2, we have to consider both the learning (i.e., identifying the best incentive method) and the tuning (i.e., identifying the best incentive in an incentive method by determining appropriate values of the parameters of the method). The algorithm proposed in Chapter 4, cannot deal with this variant. Indeed, the budget for the crowdsourcing projects corresponding to this variant will not be sufficient to explore all candidate incentives, as there are many incentives. Even if it is enough, the residual budget is unlikely to be enough to

effectively exploit the best incentive explored. Hence, in this chapter we develop another algorithm to solve the ISP2. The proposed algorithm in the chapter, takes into account the possible correlations between the incentives in a cluster to effectively and quickly identify a highly effective incentive. We first formalise the ISP2 (Section 5.1). We then develop an algorithm to deal with the ISP2 (Section 5.2). After that, we evaluate the performance of the algorithm through simulations (Section 5.3). Finally, we conclude the chapter with a summary (Section 5.4).

5.1 The Problem

In Chapter 3, we have described general descriptions of the ISP together with its two variants, the ISP1 and the ISP2. Also, in Section 4.1 we have presented the formal description of the first variant, the ISP1. Now, we formally present the second variant, the ISP2, when we keep the chosen incentive methods. That means in this variant, there are much more incentives to be selected in the learning process.

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 $v_a = (v_a^{(1)}, \dots, v_a^{(K_i)})$, where $v_a^{(k)}$ is the value corresponding to the k th parameter and $v_a^{(k)} \in [v_{min,i}^{(k)}, v_{max,i}^{(k)}]$ ($v_{min,i}^{(k)}, v_{max,i}^{(k)} \in \mathbb{R}$). Let c_a denote the cost of applying incentive a once. The expected utility of this incentive is μ_a which is unknown in advance. Let $\mathbf{N} = \{n_a^{(t)} \mid t = 1, \dots, T; a \in C_i; i = 1, \dots, C\}$ denote a policy, where $n_a^{(t)}$ is the number of times incentive a is applied in period t , i.e., incentive a is provided to $n_a^{(t)}$ different groups. Let $r_a^{(t)}$ 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 expected overall utility:

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

This model is Contribution 4 as presented in Section 1.3.

5.2 The BOIS Algorithm

As presented in Section 2.2.2, although there are a certain number of many-armed bandit algorithms that can be considered to use for the ISP2, since the problem is complex (with several characteristics), these algorithms are not efficient to deal with the ISP2. Then, in Section 2.2.3, we discussed that Bayesian Optimisation (BO) is a promising approach to the problem. In this section we introduce **Bayesian-optimisation-based Incentive Selection** (henceforth, BOIS), a novel algorithm for the ISP2. We first give an overview

of the algorithm (Subsection 5.2.1). Then, we detail how BOIS splits the learning and tuning process into steps and how it acts in these steps (Subsections 5.2.2–5.2.4).

5.2.1 Algorithm Overview

The idea of BOIS is using a MAB approach to deal with the learning problem (i.e., identifying the best cluster) and using BO 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 chooses an incentive whose value of the acquisition function corresponding to this incentive is the largest compared to those of the other incentives in all clusters. As presented in Subsection 2.2.3, two main components of BO are the surrogate model and the acquisition function. The surrogate model represents our belief on the shape of the objective function based on the observations. And the acquisition function is to help identify an appropriate point to query so as to quickly learn the best point in the objective function. Regarding the first component, we use a GP, a non-parametric Bayesian method, to learn the underlying correlation within a cluster, which is assumed to be a function. In fact, GPs are a useful tool to learn these functions (Subsection 2.2.3). Regarding the second component, since in the ISP the objective is to maximise the cumulative utility in a limited time budget, we choose EI and GP-UCB¹ to be as the acquisition function. In more detail, because EI tends towards exploitation, we skip this function. Also, as TS helps identify the next sample by sampling from the Gaussian process, it needs the time budget to be large enough which is not the case in the ISP. Thus, we do not use this function. EI and GP-UCB are designed to strike a balance between exploration and exploitation. So, we run the simulations with both of these two functions. However, as will become apparent GP-UCB performs better than EI in most cases. The reason is that UCB-based methods in general are inherently designed to not only obtain a good exploration-exploitation balance but also minimise the cumulative regret, i.e., maximise the cumulative utility (Subsection 2.2.2.1). Hence, in Section 5.3, we only present the results with the GP-UCB acquisition function, which is implemented as the UCB of the estimate of the incentive' effectiveness.

The general idea of tuning parameter values of an incentive method (i.e., finding the best incentive in a cluster) using BO is the following. In each period, based on the incentives sampled in the previous periods, BOIS estimates the mean rewards of the incentives so far 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 of the cluster. BOIS will then choose the candidate incentive in the cluster which has the highest UCB to be applied in that period.

¹See Subsection 2.2.3.2 for details of the acquisition functions.

Algorithm 4 High-level Overview of the BOIS Algorithm

Input: financial budget (B), time budget (T), number of clusters (C), ..**Predefined parameters:** $\epsilon_1, \epsilon_2, U_1, ..$ **Output:** applying policy, overall utility**Note:** See Algorithm 5 for the full detail version of the BOIS algorithm.

- | | | |
|----------------------|---|---|
| Sampling | { | <ul style="list-style-type: none"> 01: Distribute the budget for sampling ($\epsilon_1 B$) over C clusters. 02: In each cluster, apply some incentives so that each incentive has about U_1 sampled users. The number of incentives chosen is constrained by the given budget for sampling of the cluster. 03: For all incentives in each cluster, estimate the mean rewards (using GPR) and calculate the UCBs. |
| Stepped Exploitation | { | <ul style="list-style-type: none"> 04: The budget for stepped exploitation is $b_2 = \epsilon_2 b$, where b is the residual budget. 05: This budget (b_2) is distributed across T_s periods, where $T_s + 1$ is the remaining periods including the last one. 06: for each of the T_s periods do 07: Select the incentive (a) with the highest UCB in all clusters. 08: Apply a as many times as possible within the given budget. 09: Update the surrogate model of the cluster containing a (C_i). 10: Calculate the UCBs of all incentives in C_i. |
| Pure Expl. | { | <ul style="list-style-type: none"> 11: Apply the best incentive (in all clusters) with the residual budget. 12: Apply the second best incentive (in all clusters) with the residual budget. 13: Repeat the process until the residual budget is not enough to apply any incentive. 14: return the applying policy and the overall utility. |
-

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 to obtain good estimates of the effectiveness of the incentives. Hence, 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 *exploration step*. Finally, in the last period it applies the best incentive with the remaining budget. This step is called the *exploitation step*, as it simply exploits the best incentive after exploring in the previous periods. A high-level overview of the algorithm is presented in Algorithm 2.

Regarding the UCBs, as in the first periods we are not confident about the estimates of the incentives' effectiveness, 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 exploring. 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 exploring to exploiting.

In order for BOIS to choose an incentive in a period ($t + 1$), at the end of the previous

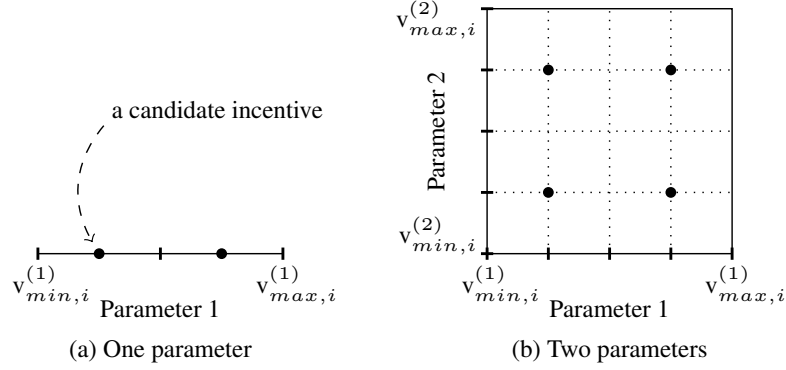


FIGURE 5.1: An illustration of candidate incentives in a cluster in the Sampling step when the cluster has one (a) and two (b) parameters

period (t), it uses Gaussian process regression to estimate the mean utilities of all incentives in each cluster. The results of the estimation are $\hat{\mu}_a^{(t)}$ and $\hat{\sigma}_a^{(t)} \forall a \in C_i; i = 1, \dots, C$. After that, it calculates the potential effectiveness of all incentives:

$$d_a^{*(t)} = \frac{1}{c_a} \left(\hat{\mu}_a^{(t)} + z^{(t)} \frac{\hat{\sigma}_a^{(t)}}{\sqrt{m_a^{(t)} g_a}} \right). \quad (5.1)$$

In Equation 5.1, $z^{(t)} = Z(1 - \frac{t-1}{T-2})$, 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 $t = T - 1$, $z^{(t)} = 0$ means that it focuses only on exploitation. Additionally, the denominator, $\sqrt{m_a^{(t)} g_a}$, signifies that the exploration level is inversely proportional to the number of sampled users.

As discussed in Section 3.1 and Section 5.1, the ISP2 corresponds to the crowdsourcing projects where the incentives in each cluster must be correlated. However, in some projects, only some clusters are correlated, the other ones are not. In these projects, we can apply BOIS by considering each incentive in uncorrelated clusters as the only incentive in a new cluster. We can also consider applying HAIS in these projects by continue choosing some incentives in correlated clusters. If we have no prior knowledge about user performance, we can choose the incentives randomly.

In the next subsections, details of the steps will be discussed. The explanations will be linked to the corresponding parts of the detailed pseudocode of BOIS shown in Algorithm 5. This algorithm helps us achieve Contribution 5 as presented in Section 1.3.

5.2.2 The Sampling Step

As mentioned above, the purpose of this step (Lines 2–11) 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 5.1. 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., $v_{min,i}^{(k)} + 0.25\Delta_i^{(k)}$ and $v_{min,i}^{(k)} + 0.75\Delta_i^{(k)}$, where $\Delta_i^{(k)} = v_{max,i}^{(k)} - v_{min,i}^{(k)}$ (Figure 5.1a). From these values, we have a set of 2^{K_i} candidate incentives to be sampled. Figure 5.1b describes 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 the budget on further exploration and exploitation. To deal with this, 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 ($\forall i = 1, \dots, C$). 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 U_1 (e.g., 10) sampled users, which is calculated by rounding the division 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). $\lfloor b_1/c_a \rfloor$ in Line 8 is to guarantee the budget being used in this step does not exceed $\epsilon_1 B$. The algorithm stops sampling when the budget for sampling is exceeded (Lines 6–7).

5.2.3 The Stepped Exploitation Step

At first, BOIS sets the budget for exploration, a specific portion of the residual budget which is identified by ϵ_2 , e.g., 0.5 (Line 12). Then, in each period (t) before the deadline, it will choose the incentive (a) with the highest potential effectiveness (Line 14). The incentives are chosen based on their UCBs which 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 which have high estimates) and exploring (choosing the incentives which have high potential to be the best incentive). This is part of the reason that UCBs are a good choice to be used as the acquisition function in this problem.

Algorithm 5 Detailed Pseudo code of the BOIS Algorithm**Input:**

$B, T, C,$ ▷ financial budget, time budget, number of clusters
 $K_i (\forall i = 1, \dots, C)$ ▷ number of parameters of each incentive

Predefined parameters:

$\epsilon_1, \epsilon_2, U_1, D_{min},$ and Z ▷ see Table 5.2 for their descriptions

Output:

$r, \mathbf{N} = \{n_a^{(t)} \mid t = 1, \dots, T; a \in C_i; i = 1, \dots, C\}$ ▷ overall utility, applying policy

Note: $\text{ApplyIncentive}(i, n)$ 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:      $a \leftarrow \text{NextSample}(C_i);$ 
06:     if  $b_1 < c_a$  then
07:       | Stop the for and while loops;
08:        $n_a^{(1)} \leftarrow \max\{1, \min(\lfloor U_1/g_a \rfloor, \lfloor b_1/c_a \rfloor)\};$ 
09:        $r_a^{(1)} \leftarrow \text{ApplyIncentive}(a, n_a^{(1)});$ 
10:        $b_1 \leftarrow b_1 - n_a^{(1)} c_a; b \leftarrow b - n_a^{(1)} c_a;$ 
11:  $\text{UpdateEstimates}(C_i, 1, Z) \quad \forall i = 1, \dots, C;$ 

Stepped Exploitation {
12:  $b_2 \leftarrow \epsilon_2 b;$  ▷ residual budget for exploration
13: for  $t = 2 \rightarrow T - 1$  do
14:    $a \leftarrow \text{argmax}_{a' \in C_i; i=1, \dots, C} \{d_{a'}^{*(t-1)}\};$ 
▷ select an incentive that maximises the acquisition function
15:   if  $d_a^{*(t-1)} < D_{min}$  then ▷  $a$  is too bad
16:     |  $i \leftarrow$  a random cluster;
17:     |  $a \leftarrow$  a random incentive in  $C_i;$ 
18:   if  $b_2 < c_a$  then ▷ budget for exploration is exceeded
19:     | Stop the for loop;
20:    $n_a^{(t)} \leftarrow \max\{1, \min(\lfloor U_1/g_a \rfloor, \lfloor b_2/c_a \rfloor)\};$ 
21:    $r_a^{(t)} \leftarrow \text{ApplyIncentive}(a, n_a^{(t)});$ 
22:    $b_2 \leftarrow b_2 - n_a^{(t)} c_a; b \leftarrow b - n_a^{(t)} c_a;$ 
23:    $\text{UpdateEstimates}(C_i, t, Z);$ 
▷ update the surrogate model  $(\hat{\mu}_{a'}^{(t)}$  and  $\hat{\delta}_{a'}^{(t)}) \quad \forall a' \in C_i$  where  $a \in C_i$ )

Pure Expl. {
24: while  $b \geq \min_{a \in C_i; i=1, \dots, C} c_a$  do
25:    $a \leftarrow \text{argmax}_{a' \in C_i; i=1, \dots, C} \{d_{a'}^{*(T-1)}\};$ 
26:    $n_a^{(T)} \leftarrow \max\{1, \lfloor b/c_a \rfloor\};$ 
27:    $r_a^{(T)} \leftarrow \text{ApplyIncentive}(a, n_a^{(T)}); b \leftarrow b - c_a n_a^{(T)};$ 

28:  $r \leftarrow \sum_{t=1}^T \sum_{i=1}^C \sum_{a \in C_i} r_a^{(t)};$  ▷ overall utility
29: return  $r, \mathbf{N};$ 

```

Algorithm 6 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 5.1 $\quad \forall a \in C_i$;

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 15–17). Note that 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 you can set D_{min} to this value or even 10 if we are not quite sure about this number. 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 U_1 sampled users (Lines 20–21). This step stops when the budget for exploration is exceeded (Lines 18–19).

5.2.4 The Pure Exploitation Step

This step (Lines 25–27) simply applies the best incentive explored with the remaining budget. From Equation 5.1 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 expected effectiveness.

5.3 Experimental Evaluation

To systematically evaluate the performance of BOIS, we use 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 that we have explored the main cases in a comprehensive fashion. In the following, we present the benchmarks (Subsection 5.3.1), the experimental settings (Subsection 5.3.2), the corresponding results (Subsection 5.3.3), and practical usage of the BOIS algorithm based on the results (Subsection 5.3.4). This section corresponds to Contribution 6 as presented in Section 1.3.

5.3.1 Benchmarks

As presented in Section 2.2, there are no algorithms in the literature what can be used to solve the ISP2. Specifically, state-of-the-art algorithms are not specifically designed to deal with choosing the best cluster together with tuning its parameter values. So, we choose some of them as the benchmarks and make a number of modifications for them to perform well with the ISP2.

(1) **H AIS**. The algorithm is described in detail in Section 4.2. When applying **H AIS** into the ISP2, it first randomly chooses a certain number of incentives. Then, it applies these incentives so that it has about U_1 sampled users on each incentive. The number of incentives chosen is identified by $\epsilon_1 B$. After that, **H AIS** deals with the chosen incentives as described in Algorithm 3. Although both **H AIS** and **BOIS** are to deal with the ISP, they are to solve different variants of the problem. **H AIS** is for the ISP1 where the incentives are uncorrelated, while **BOIS** is for the ISP2 where the incentives are correlated. Hence, we run **H AIS** as a benchmark for **BOIS** is to evaluate the performance of **H AIS** on the ISP2 in comparison with **BOIS**.

(2) **ϵ -first**. This algorithm (Section 2.2.2.1) spends $\epsilon_1 B$ (where ϵ_1 is specified in advance, e.g., 0.2) in the first period to explore by sequentially applying a random incentive in each cluster until this budget is exceeded. With a chosen incentive a , it applies this incentive $\max\{1, [U_1/g_a]\}$ times to obtain about U_1 (e.g., 10) sampled users. In the second period, it uses Gaussian process regression to estimate the best incentive. Then it spends the subsequent period purely exploiting the best incentive with the remaining budget, i.e., $(1 - \epsilon_1)B$.

(3) **ϵ -BOIS**. This is **BOIS** but the exploration step is from **ϵ -first**. Specifically, in the first period, this algorithm does exactly the same as in **ϵ -first**, that is to use $\epsilon_1 B$ for sampling incentives randomly. In the next periods (except the last one), it applies the stepped exploitation step of **BOIS** (Section 5.2.3). And in the last periods, the algorithm conducts pure exploiting as in **BOIS** (Section 5.2.4). Although **ϵ -BOIS** is a combination of **BOIS** and **ϵ -first**, the second step (stepped exploitation) is the main idea of **BOIS** and more complicated than the first step (which is borrowed from **ϵ -first**). Therefore, we name it **ϵ -BOIS**. The purpose of running this algorithm is to examine the effectiveness of the stepped exploitation step of **BOIS**. For this reason, we do not consider it as a state-of-the-art algorithm.

(4) **Decaying ϵ -greedy** (henceforth, **ϵ -greedy**). This algorithm spreads the budget B over T periods. In each period, with the given budget, it applies the current best incentive with probability $(1 - \epsilon)$ and a random incentive in a random cluster with

probability ϵ , where $\epsilon = (T - t)/(T - 1)$. When $t = 1$, $\epsilon = 1$ means that it totally explores. When t increases, ϵ gradually decreases. And when $t = T$, $\epsilon = 0$ means it completely exploits the best incentive explored. At the end of each period except the last one, it uses Gaussian process regression to estimate the best incentive for the next period.

(5) Random. Similar to ϵ -greedy, this algorithm spreads the budget B over T periods. Then in each period, it simply applies a random incentive in a random cluster with the given budget.

(6) Optimal. It simply applies the real best incentive all the time. To have the optimal solution, we have to know the expected values μ_a ($\forall a \in C_i; \forall i = 1, \dots, C$) in advance, which is typically impossible in practice. Therefore, this approach represents an upper bound of what any algorithm could achieve.

5.3.2 Simulation Settings

To evaluate the performance of the algorithms, we run simulations in three different settings where the independent variables are financial budget, time budget, and number of clusters. For each value of the independent variable, we run 1,000 simulations to achieve statistically significant results at the 99% confidence level. Error bars of the line graphs in Figures 5.3–5.4 represent the confidence intervals. The simulations run in this chapter are different from the ones in Chapter 4 is that, in this chapter, the candidate incentives are grouped in clusters and the incentives in each clusters are correlated, whereas in Chapter 4, all candidate incentives are not correlated.

Next, in Subsection 5.3.2.1, we detail the ranges of the parameter values used for randomisation in the simulations. Then, in Subsection 5.3.2.2, we detail the values of the predefined parameters of the algorithms.

5.3.2.1 Ranges of the Quantities for Randomisation

In the simulations of each setting, the related parameters, i.e., B , T , C , group sizes, utility, and the amount of prize money for the best user (except the corresponding independent variable) are generated randomly in specific ranges. The ranges of these parameters are chosen to represent realistic settings in real crowdsourcing projects and are described in Table 5.1. Ranges of some of the parameters are already discussed in Section 4.3.2.1, and hence we do not discuss here.

Regarding the *number of clusters*, we choose 10 as the maximum value of C . That means we can choose up to 10 incentive methods. But, in reality, we believe this number is

TABLE 5.1: 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 clusters | C | 1 | 10 | Cluster |
| Group sizes | g_a | 3 | 50 | User |
| Real densities | δ_a | 30, 60 ^(*) | 90 | Utility per £ |
| Utility stand deviations | σ_a | $0.2\mu_a$ | $0.6\mu_a$ | - |
| The amount of prize money for the best user | - | 1 | 25 | £ |
| Financial budget | B | 10 | 200 | Times of the round cost (**) |
| Time budget | T | 2 | 30 | Period |

(*) *Densities of the worst incentives in each cluster are 30, while the density of the best incentive in the worst cluster is 60 utility per £.*

(**) *Round cost is the cost of applying all the clusters, where in each cluster the centre incentive is applied such that it has about U_1 users.*

likely to be small, such as 3, 2, or even 1. When we have one more cluster, the budget should be large enough to cover the other sampled incentives. Also, the chance of identifying the best cluster decreases. Indeed, if in a crowdsourcing project, we choose 10 clusters, with the average values of other quantities (i.e., the number of parameters is 2, the group size of about 30, and the average payment for a user is £1.00) and two sampled incentives in each parameter of a cluster, the cost for initial estimates is about £1,200. Also, the overall budget should be large enough so that the algorithms can use the remaining budget to exploit the best incentive explored. Based on the insights from Subsection 4.3.4, the overall budget should be at least about 10 (ideally 20) times the cost of initial estimates. Based on this, the budget for this project should be at least about £10,200 (ideally £20,400). These amounts of money are large for a crowdsourcing project. Hence, we keep 10 clusters as the maximum in the simulations.

Regarding the *amount of prize money for the best user*, we choose £25 as the maximum value. In the simulations, every user participating in a contest will be paid a certain amount of money (£0.50) as the base payment. So, we have to pay £25 for a group of 50 users (the maximum value of the group size). Also, we want to keep the total spend on the contests always less than or equal to that of individual-based incentive methods (such as paying for performance or bonuses), which is about £1 for each user. Thus, the maximum amount of prize money is £25.

5.3.2.2 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 of ϵ -first, we first run this algorithm with different values (such as 0.05, 0.10, 0.20, 0.30, and 0.40). Then we choose

TABLE 5.2: Values of the Algorithms' Predefined Parameters in the Simulations.

| Algorithm | Parameter | Value | Description |
|-------------------|--------------|-------|--|
| BOIS | ϵ_1 | 0.10 | Budget limit for sampling. |
| | ϵ_2 | 0.50 | Budget for stepped exploitation (calculated based on the residual budget). |
| | U_1 | 20 | Target number of sampled users to obtain after the first (sampling) period. |
| | D_{min} | 40 | The minimum utility which help eliminates unacceptable incentives (whose effectiveness is below this value). |
| | Z | 1.96 | Critical value corresponding to the confidence level in eliminating ineffective incentives (i.e., to calculate confidence intervals of the incentives' estimates). |
| HAIS | ϵ_1 | 0.10 | Budget limit for exploration. |
| | ϵ_2 | 0.50 | Budget for stepped exploitation (calculated based on the residual budget). |
| | 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 for exploration. |
| | U_1 | 20 | Target number of sampled users to obtain after the first (sampling) period. |
| ϵ -BOIS | ϵ_1 | 0.10 | Budget for exploration. |
| | ϵ_2 | 0.50 | Budget for stepped exploitation (calculated based on the residual budget). |
| | U_1 | 20 | Target number of sampled users to obtain after the first (sampling) period. |

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 5.3.3 we only present the results on the simulations with the following values of the algorithms' predefined parameters as described in Table 5.2. Regarding the predefined parameters of BOIS and ϵ_2 of ϵ -BOIS, as most of them are self-explanatory and some of them are already discussed in Subsection 5.2.1, we do not explain them here.

For each value of the independent variables, we run 1,000 simulations to achieve statistically significant results at the 95% confidence level. Error bars of the line graphs in

Figures 5.3–5.6 represent the confidence intervals.

5.3.3 Results

In general, BOIS performs the best in most cases except when the number of clusters is larger than 50 (Figures 5.2–5.4). This is acceptable, as in a real crowdsourcing project, the number of incentive methods (i.e., clusters) to be learned is not often that high. In general, the performance of BOIS is up to 92% of the optimal solution and up to 48% better than state-of-the-art benchmarks. The reason that BOIS' high performance is that (1) it makes use of the periods before the deadline to gradually determine the best incentive (by using BO), and (2) it has a good sampling method (miniMax space-filling design). In the next subsections, we will discuss these two issues.

However, as the purpose running HAIS as a benchmark is to see how an algorithm designed for the uncorrelated ISP to be applied into the correlated ISP compared to BOIS. With this purpose, we focus more on the comparison between BOIS and HAIS than the other algorithms. So, we discuss the performance of BOIS before other algorithms. In general, HAIS does not perform well and its performance is worse than that of BOIS in most cases. This is because HAIS does not make use of the possible correlations between incentives in a cluster to quickly and effectively learn the best incentive of the cluster. These results say that it is better to use BOIS instead of HAIS to deal with the correlated ISP (the ISP2).

5.3.3.1 Taking Advantage of the Time Budget

Figure 5.2 shows that with a larger time budget, the algorithm performs better, especially when T is larger than about 15. Specifically, after the sampling step conducted in the first period, the performance of BOIS increases significantly from a utility of about 70 per \mathcal{L} at the end of the first period up to about 80 per \mathcal{L} when $T = 15$. This indicates that Bayesian optimisation can quickly approach a global optimum (i.e., the real best incentive). Figure 5.3 presents a clearer view of the effectiveness of the stepped exploitation step. Concretely, when the budget becomes larger but still small (less than about £3,000), the performance of both ϵ -first and ϵ -BOIS improves significantly. However, when the budget is larger than £3,000, ϵ -first performs slightly better as it has more budget for exploration, whereas ϵ -BOIS continues improving its performance clearly. This shows the fact that ϵ -BOIS makes use of the budget to continue exploring the incentives by conducting stepped exploitation. The reason ϵ -first does not perform well with larger budgets is that the way it explores is inflexible (i.e., always $\epsilon_1 B$). Indeed, when B is small, the budget for exploration is not enough, so that the Gaussian process regression conducted in the second period does not have enough samples to identify one of the best incentives. Figure 5.3 also suggests that the larger the budget the better.

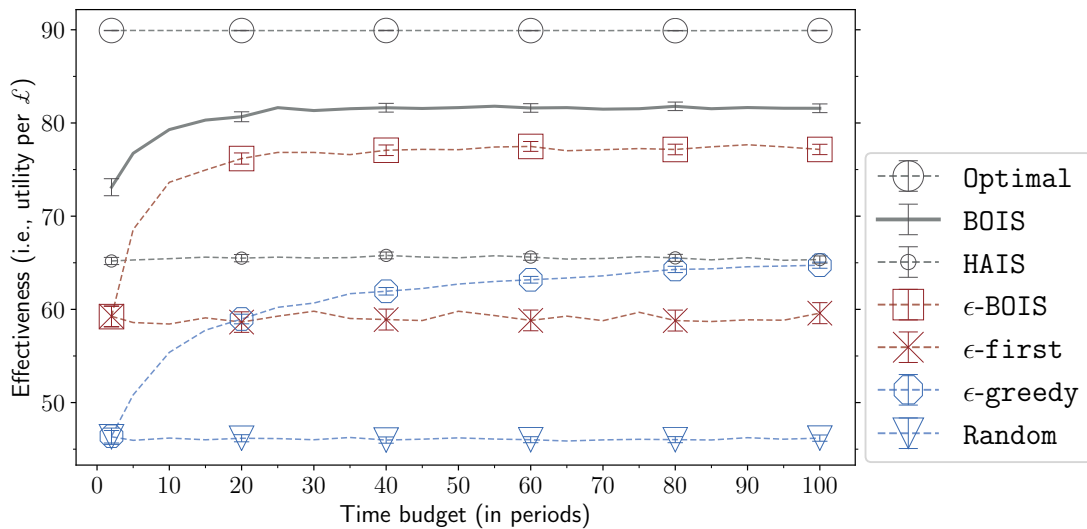


FIGURE 5.2: Performance of Algorithms for Different Time Budget Values

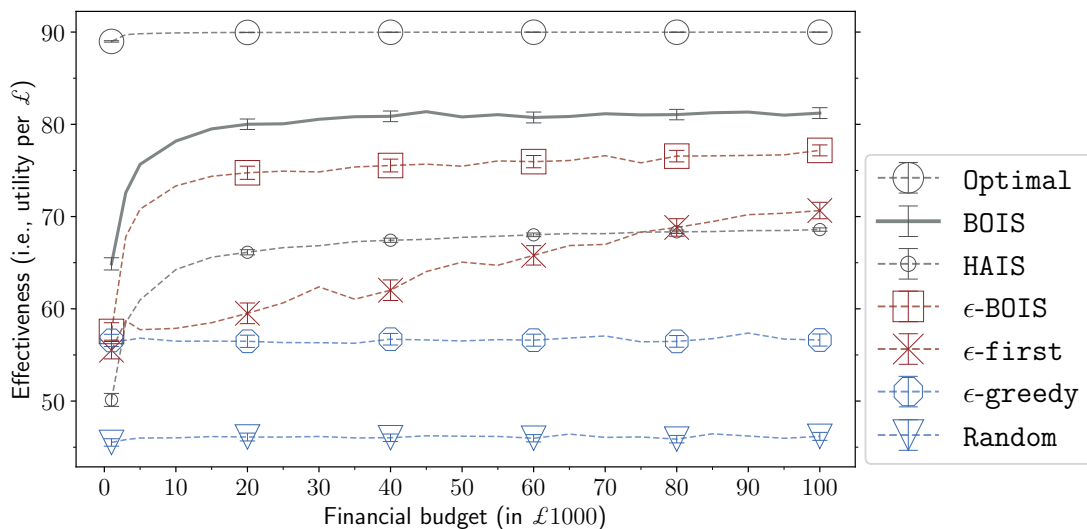


FIGURE 5.3: Performance of Algorithms for Different Financial Budget Sizes

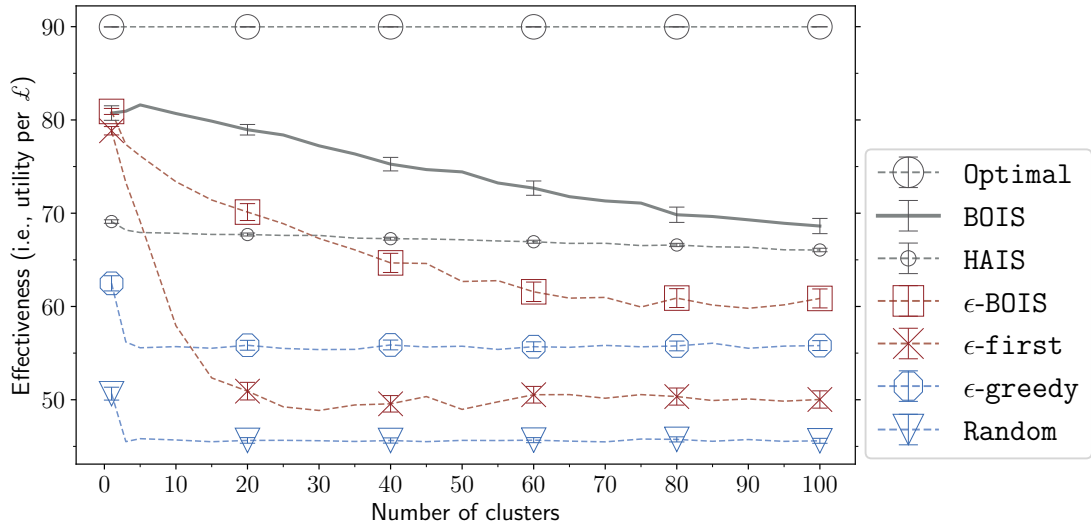


FIGURE 5.4: Performance of Algorithms for Different Numbers of Clusters

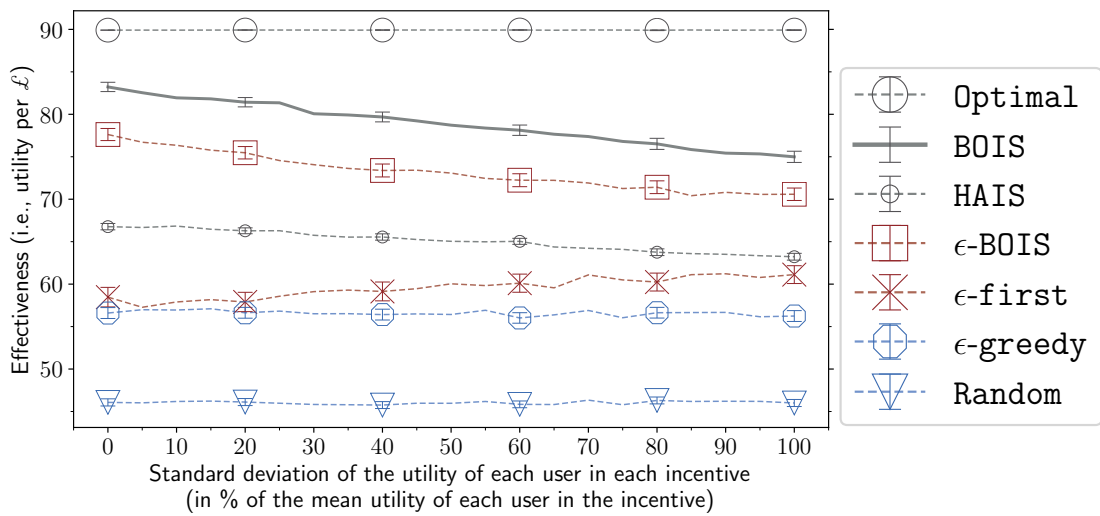


FIGURE 5.5: Performance of Algorithms for Different Values of the Standard Deviation of the Incentives' Utilities

This is due to the stepped exploitation as it helps spread the budget across the periods effectively to identify the best incentive.

With a similar trend, ϵ -greedy also performs better with larger T , as it has more time to explore. Yet, its performance is far below that of BOIS and ϵ -BOIS. This is because it does not have a good exploration strategy. Actually, conducting exploration by randomly selecting an incentive does not make use of existing estimates of the incentives so far as in the stepped exploitation step.

5.3.3.2 Effective Sampling

Figure 5.2 shows that BOIS' sampling step is effective. Indeed, when $t = 2$ (i.e., no stepped exploitation executed), BOIS enjoys much higher utility than ϵ -BOIS (about a utility of 70 compared to just about 60 per £). However, in order for the miniMax space-filling design to be effective, the financial budget should be large enough to sample all 2^{K_i} candidate incentives $\forall i = 1, \dots, C$. This is why BOIS' performance increases significantly when the budget is in the range from £500 to about £4,000 in Figure 5.3. We can also see this in Figure 5.4. More specifically, the performance of BOIS drops significantly when the number of clusters becomes larger. This is because with a fixed budget (B) and a larger number clusters, $\epsilon_1 B$ is not enough to sample all the candidate incentives in all clusters.

Similar to Figure 5.4, regarding the number of parameters, as the algorithm does not scale well to settings with large values of K_i , we only run with $K_i = 2, 3$. Their results have a similar trend as in Figure 5.4, that BOIS performs well when $K_i = 2$ (a utility of about 80 per £), but then its performance gradually drops down to about 74 when $K_i = 3$ and 65 when $K_i = 4$. Also, when $K_i = 4$, the time to run the algorithm is less than 2 minutes, which is likely to be acceptable in most practical applications.. Additionally, increasing the number of parameters (e.g., from 3 to 4) results in increasing the number of incentives exponentially. Therefore, in real crowdsourcing projects, it is not effective to have more than 4 parameters as it requires an extremely large budget to sample the incentives in each cluster.

5.3.4 Practical Usage of the BOIS Algorithm

The above-mentioned results suggest several guidelines for using this algorithm effectively in practice. Most of them are similar to the ones related to the usage of HAIS. Hence, we just name or briefly described the ones which are already discussed in Subsection 4.3.4.

First, the larger the budget the better (Figure 5.3). This figure also suggests that the budget should be large enough (e.g., at least £2,000 as in the simulations) for BOIS

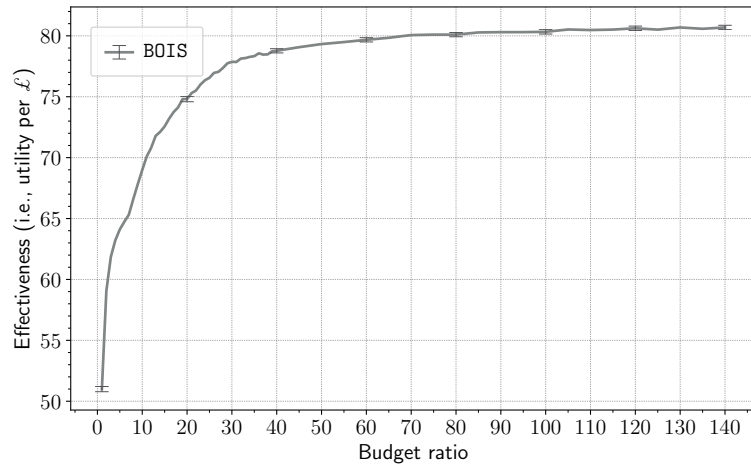


FIGURE 5.6: Performance of BOIS for Different Budget Ratios. Budget ratio is $B/\text{round_cost}$.

to achieve a good performance. BOIS needs enough budget not only to sample all the incentives according to the miniMax space-filling design, but also to conduct stepped exploitation and then purely exploiting the best incentive explored. Second, the time budget should be large enough (e.g., 10 periods), but does not need to be very large (e.g., 50 periods) so that BO has enough time to determine the best incentive (Figure 5.2).

Third, the fewer clusters and parameters the better (Figure 5.4). In order to help identify a reasonable number of clusters, we run other simulations to investigate BOIS' performance with different ratios of the financial budget compared to the round cost. The result is shown in Figure 5.6. This figure is consistent with Figure 5.3b, that the larger the budget is the better BOIS performs. It can be seen from this figure that, the performance of BOIS increases significantly when the budget ratio is from 1 to about 20. After that, it still improves, but slowly. This suggests that, the round cost should be at least 20. Based on this, with a given financial budget, we can easily determine an appropriate maximum number of clusters. If there are many (e.g., 15) candidate clusters to choose from and the budget is not large enough (e.g., just about 5 times the round cost), 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 other parameters.

5.3.5 When to use HAIS and BOIS Algorithms to deal with the ISP

We have built two algorithms to deal with two variants of the ISP. They are HAIS (Chapter 4) for the uncorrelated ISP (the ISP1) and BOIS (this chapter) for the correlated ISP (the ISP2). We have also run experiments of both algorithms for both variants (the ISP1 and ISP2) to see how the two algorithms perform in each variant of the ISP. The results (Subsections 4.3.3 and 5.3.3) show that it is better to use each algorithm for the

variant that it is designed for. This is because each algorithm can take more advantage of the characteristics of the corresponding variant to efficiently solve the ISP. Specifically, **H AIS** is better for the uncorrelated ISP, while **B OIS** is better for the correlated ISP. With the ISP1, **H AIS** takes advantage of the characteristic that the number incentives is usually not large (compared to that of the correlated ISP) and the incentives are uncorrelated to simultaneously explore highly effective incentives in the Hoeffding step (right in the second period). Exploring like this when dealing with the ISP2 is not effective as the number of incentives is usually large. It might spend a significant amount of the budget for exploration. Thus, the residual budget is not enough to effectively exploit the best incentives explored. Similarly, with the ISP2, **B OIS** makes use of the possible correlations between the incentives in each cluster to quickly identify the best incentive in each cluster. Because of this, **B OIS** can find the best incentive among clusters more quickly and efficiently than **H AIS**.

Regarding the mixed-correlated ISP (the ISP3), as discussed in Section 3.2.3, we can solve this problem directly by building an algorithm for this variant or indirectly by using designed algorithms for the other two variants. The direct way is left for future work. So, we can use either **H AIS** or **B OIS** to deal with the ISP3 by considering the budget ratio, which is the ratio of the budget of a crowdsourcing project to the round cost. As discussed earlier in Section 5.3.4, as when the budget ratio is from 1 to 20 the performance of **B OIS** increases significantly. After that the algorithm still improves but much more slowly. That means when the budget for a project is large enough (i.e., larger than 20 times the round cost), we can apply **B OIS** for this project. Note that when applying **B OIS** for a mixed-correlated project, each uncorrelated incentive is considered as the only incentive in a cluster. When the budget is less than 20 times the round cost, before using **B OIS** we should continue choosing candidate incentives in some clusters so that the budget ratio is at least 20. In cases we did this with all clusters but the ratio is still less than 20, we should consider using **H AIS** as this time we do not have any correlated clusters. As discussed earlier, **B OIS** does not perform well (compared to **H AIS**) in these cases.

5.4 Summary

In this chapter, we presented the ISP2 (or correlated ISP), a variant of the ISP where the candidate incentives are correlated. We then developed **B OIS**, an adaptive algorithm, to solve the ISP2 effectively. The ISP2 is a combination of the learning (the optimal cluster) and tuning (the corresponding parameters) problems. So, the **B OIS** algorithm combines two different techniques in a sequential decision process to deal with the two problems at the same time. Actually, to deal with the learning problem, in a period, it uses a MAB approach, which simply selects the most promising incentive in the most

promising cluster. To tackle the tuning problem, the algorithm uses Bayesian optimisation together with Gaussian process regression, which makes use of possible correlations between the incentives in a cluster to determine the most promising incentive effectively. The performance of the algorithm was then evaluated by a series of simulations. Specifically, we run the simulations with different financial budget sizes, different time budget values, and different numbers of clusters. The results of the simulations show that BOIS performs the best in most cases. Throughout this chapter, we obtain the first three contributions as presented in Section 1.3. Specifically, the model (i.e., the formal presentation of the ISP presented in Section 5.1) is Contribution 4, the BOIS algorithm (detailed in Section 5.2) is Contribution 5, and the evaluation of the algorithm's performance (Section 5.3) is Contribution 6.

Similar to HAIS, BOIS helps us achieve the first three objectives of the research as mentioned in Section 1.2, as it is efficient (**RO1**), autonomous (**RO2**), and adaptive (**RO3**). Moreover, together with HAIS, BOIS helps us attain the last research objective (**RO4**), in which the solution is complete (i.e., can be applied to any crowdsourcing project). Indeed, with these two algorithms, we can solve the ISP in crowdsourcing projects where the candidate incentives are uncorrelated or correlated. As discussed in Subsection 3.2.3, we can apply HAIS or BOIS to the projects with mixed-correlated incentives, i.e., projects where the incentives in some clusters are correlated while the incentives in the other clusters are uncorrelated). Depending on how much prior knowledge we have about the performance of users in the projects and how large the financial budget is, we can choose to go with HAIS or BOIS. More specifically, we can consider using HAIS if the budget is not large compared to the chosen candidate incentives. We can consult Subsection 4.3.4 to have a good decision on how large of the budget is enough compared to the candidate incentives. And, we can consider using BOIS if the budget is large (compared to the chosen candidate incentives) and we have no or very little prior knowledge about user performance. We can consult Subsection 5.3.4 to have a good decision on how large of the budget is enough compared to the incentives in the chosen incentive methods.

Chapter 6

Conclusions and Future Work

This final chapter summarises our solution to the incentive problem and outlines opportunities for future work. Specifically, in Section 6.1, we summarise the main body and results of this thesis while, in Section 6.2, we discuss several ways that can advance our work.

6.1 Summary

Microtask crowdsourcing is an efficient tool to complete small tasks which are difficult or impossible for computers but easy for humans. To run a crowdsourcing project, the requester has to not only attract enough users (i.e., contributors) to the project, but also have a good way to encourage users to participate in the project and to perform tasks with a high quality. There are extensive studies of motivation and incentives from many fields of study, such as psychology, sociology, organisational behaviour, marketing, and computer science, under various perspectives, such as human in general, workers in firms, customers of brands, members social in groups or online communities, and contributors in crowdsourcing systems. These studies suggest a variety of candidate incentive methods that can be offered in specific crowdsourcing projects. In detail, incentive methods might have different sets of parameters with different ranges of the parameters. An incentive corresponds to specific values of the parameters. However, the effectiveness of these incentives are usually unknown in advance due to specific characteristics of those projects. Furthermore, many crowdsourcing platforms, such as Amazon Mechanical Turk, Clickworker, and Figure Eight, provide APIs for the requesters to manage the tasks and the submissions in an autonomous manner. This makes it possible to build autonomous agents to monitor and adaptively switch incentives when appropriate.

Against this background, instead of investigating a particular incentive, in this thesis, we focus on a more practical way to solve the incentive problem, which we call the incentive

selection problem (ISP). Concretely, we find an appropriate way for an autonomous agent to select an effective incentive (among the candidate incentives) in a microtask crowdsourcing project. In more detail, after choosing some candidate incentive methods (i.e., clusters) by using existing studies and prior knowledge about user performance in the project, the requester (or the agent) has to identify the best values of the parameters in each incentive method. The ISP has several characteristics which have not been investigated all together in the literature. They are batched pulling, budget constraints (on both finance and time), and especially the group-based nature of the incentives (or arms). In order to solve the ISP effectively, we proposed two algorithms for two variants of the problem that applied in different crowdsourcing projects, the ISP1 (or uncorrelated ISP) and the ISP2 (or correlated ISP). The first variant, the ISP1, is when the requester has prior knowledge about the performance of users or when the financial budget is small in comparison to the chosen candidate incentives. In this variant, the requester has to (or are able to) choose several incentives in each cluster before asking the agents to select the best one. The second variant, the ISP2, is when the requester has a large financial budget and no (or very little) prior knowledge about the user performance. So, they are not confident to identify highly effective incentives in a cluster, and thus, the agent have to face both the selecting (the best cluster) and tuning (the parameter values) problems.

Regarding the ISP1, we proposed **H AIS** (Chapter 4), an algorithm to help the agents efficiently select the best incentive among several candidate incentives. **H AIS** splits the selection process into four steps: sampling, Hoeffding, stepped exploitation, and pure exploitation. The sampling step happens in the first period, where the algorithm applies each incentive several times to have initial estimates of the incentives' effectiveness. At the end of this step, based on the confidence bounds of the estimates, **H AIS** eliminates clearly ineffective incentives. In the second step, the algorithm uses Hoeffding's inequality to adaptively determine how much exploration is sufficient in identifying the best incentive with a certain level of confidence. Then, in the next periods (except the last one), it conducts stepped exploitation by spreading the remaining budget over the periods and applying the current best incentive with the given budget following by an update to the incentive's estimate. Finally, in the last period, it simply applies the best incentives with the residual budget. Through extensive simulations, **H AIS** is shown to outperform state-of-the-art approaches such as **Exp3**, **ϵ -first**, **fKUBE**, and **SOAAv**. In more detail, 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. We also run other simulations to better understand the behaviour of each approach. Based on the results of the simulations, we extracted several points which are useful when applying **H AIS** in practice, such as the more budget the better (at least about 20 times the round cost), the looser deadline the better (but not necessarily too loose), and the less candidate incentives the better.

Regarding the ISP2, we proposed **BOIS** (Chapter 5), an algorithm to help agents effectively select the best cluster and also identify appropriate values of the parameters. **BOIS** uses Bayesian Optimisation (BO) with Gaussian process prior to solve the tuning problem. Additionally, it considers the selecting problem as a MAB problem. By combining MAB and BO approaches, the algorithm solves the two problems in a single process. In detail, **BOIS** spends the first period to obtain initial estimates of the small number of incentives in each cluster. Then, in each of the following periods (excluding the last one), the algorithm offers the incentive which has the largest upper confidence bound (i.e., the value of the acquisition function). After having the user performance of the offered incentive, it updates the probabilistic (i.e., surrogate) model of the utility function to repeat the same process in the next period. In the last period, it simply applies the best incentive identified. Results from our simulations show that **BOIS** performs better than state-of-the-art approaches such as ϵ -first and ϵ -greedy. More specifically, we run simulations in four different settings, where the independent variables are financial budget, time budget, number of clusters, and standard deviation of the incentives' utilities. The guidelines extracted from the simulations are consistent with those in **H AIS**. In more detail, the more budget the better (at least about 50 times the round cost), the looser deadline the better, and the less clusters the better.

When taken together, **H AIS** and **BOIS** help us achieve all the objectives of the research described in Section 1.2. Firstly, the two algorithms are shown to be efficient (**RO1**) as they perform better than other benchmarks in most cases. Secondly, both **H AIS** and **BOIS** are totally autonomous (**RO2**). The requesters just give them the candidate incentives and tell them the financial and time budgets. The algorithms will apply appropriate incentives in each period so as to maximise the total utility. Thirdly, the proposed algorithms are adaptive (**RO3**). Indeed, although the algorithms have a number of predefined parameters (as shown in Tables 4.2 and 5.2), these parameters are self-explanatory. In particular, the requesters do not need to tune them when applying the algorithms to specific crowdsourcing projects. One exception is the predefined parameter D_{min} of **BOIS**. However, as discussed in Section 5.2.3, it is not difficult to identify an appropriate value for D_{min} . Fourthly, as each algorithm deals with a variant of the ISP, the two algorithms together help us solve the ISP completely, i.e., can be applied in any crowdsourcing projects (**RO4**). Regarding this research objective, as discussed in Section 5.2.1, the two algorithms can be applied in projects with mixed-correlated incentives, i.e., projects where the incentives in some clusters are correlated while the incentives in the other clusters are not correlated. Depending on how much prior knowledge we have about the performance of users in the projects and how large the financial budget is, we can choose to go with **H AIS** or **BOIS**. More specifically, we can consider using **H AIS** if the budget is not large compared to the chosen candidate incentives. And, we can consider using **BOIS** if the budget is large (compared to the chosen candidate incentives) and we have no or very little prior knowledge about user performance. We can consult Subsections 4.3.4 and 5.3.4 to have a good decision on how large of the budget

is enough compared to the candidate incentives. That means, with the two proposed algorithms, we can deal with all crowdsourcing projects.

6.2 Future Work

The work in this thesis is an initial attempt towards solving the ISP and a number of challenges remain.

First, we have two algorithms to deal with the two variants of the ISP where the candidate incentives are uncorrelated or correlated. And, as discussed in Section 5.2.1, in crowdsourcing projects where some clusters are uncorrelated and other clusters are correlated, we can use either use HAIS or BOIS. Yet, we have not yet evaluated the performance of the two algorithms in these projects. However, we can argue that, when HAIS is used, with correlated clusters (clusters where the incentives are correlated), if we continue choosing some incentives (randomly or with a low confidence that they are good), we might miss highly effective incentives. Similarly, when BOIS is used, its simple exploration mechanism might not be efficient to explore incentives in uncorrelated clusters, whereas HAIS might do better. Therefore, in these projects, a mixed strategy that considers both types of clusters (uncorrelated and correlated) might be more efficient.

Second, our current models (the ISP1 and ISP2) assume that time periods 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 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 algorithms. Additionally, this could improve the overall performance, as the algorithms have more time to conduct stepped exploitation, especially when the time budget is limited.

Third, the models assume that the cost of pulling an arm is the same at all times. This may be limiting in more general settings. For example, some incentive methods are inherently designed with variable payment such as paying for performance (Mason and Watts, 2010) or using bonuses (Yin and Chen, 2015). 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. We have a workaround to use these methods as fixed incentives as described in Section 4.1. However, this might not take full advantage of the effectiveness of the methods. Therefore, expanding the model to cover the case of variable costs of pulling

an arm will provide requesters with more options in choosing incentives to be used in the ISP.

Fourth, currently the performance of a group is assumed to be linearly proportional to the group size as described in Section 4.3.2.3. This is due to the fact that very few studies investigate the group performance in microtask crowdsourcing. In reality, this might be more complicated. A better understanding on how people perform in groups with different sizes could help designing more realistic algorithms to solve the ISP. To do this, we need to run experiments on real microtask crowdsourcing projects. We can run this on existing platforms with a large pool of workers such as Amazon Mechanical Turk, Clickworker, or Figure Eight. We can also build our own systems and then deploy the projects as in [Ramchurn et al. \(2013\)](#). The former might be easier as we do not need to build the whole system, we only need to build the tasks. Also, it might be quicker since we do not need to recruit users. On the other hand, the latter seems to take more time for preparation and deployment. However, it is more flexible as we are not restricted to the provided APIs from a crowdsourcing platform. That means, we have more control on running controlled experiments so that we can better understand the performance of users in groups in a crowdsourcing project.

Appendix A

User Motivation Theories

There are many established theories in different fields of study (such as psychology, social psychology, sociology, economics, and organisational behaviour) related to this research. Actually, many studies have used these theories to deal with problems related to user motivation and incentives. As presented in Section 1.1, in this thesis, we do not directly solve the incentive problem. Instead, we solve the incentive selection problem. In order to do this effectively, requesters in crowdsourcing projects need a means to easily choose good candidate incentives. Yet, as there are numerous incentives studied in the literature and also used in practice, in a specific crowdsourcing project, it is not easy for the requester to choose good candidates. For this reason, we built the literature map so as to help the requesters in this task. Established theories about user motivation and incentives play an important role in building the map. This appendix presents these studies.

For a better presentation, the theories are presented in four groups corresponding to four aspects related to user motivation: the crowd, the tasks, the requester, and the user himself. Figure A.1 shows the positions of these theories in relation to the groups. These theories help us identify important design objectives (i.e., performance rewarding or task autonomy), useful incentives to enhance some motivational aspects (e.g., using quota/bonus rewarding or providing users with freedom to switch tasks rather than only do tasks provided), and causal relationships between incentives and design objectives. They also help us identify the relationships between aspects of user motivation and user behaviours. These theories are mentioned separately in some other places of the thesis, thus we present them in this appendix.

In terms of the *user*, self-determination theory (SDT) and flow theory are two well-established studies in psychology about general motivations of humans for choosing and carrying out some activities. They focus on how to have an optimal experience when conducting these activities (flow theory) and on the classification of motivations (SDT). Another well-established theory is goal-setting theory. It explains factors connected to

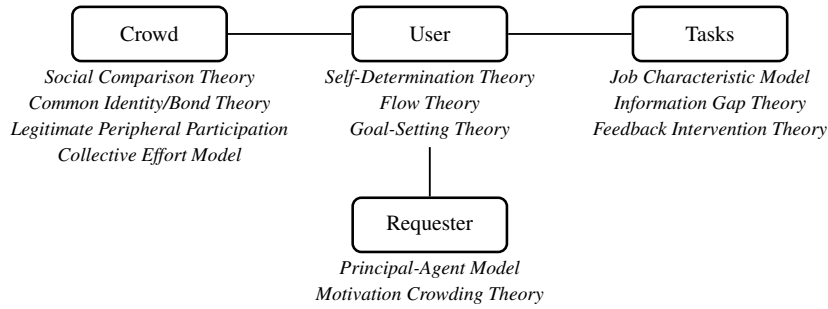


FIGURE A.1: User Motivation Theories used in building the Incentive Map. *User* is a member of the crowd.

goals that have strong influence on humans. Thus, we can motivate people by setting or helping them to set appropriate goals. In terms of the *tasks*, the job characteristic model (JCM) is also a theory that is widely used in organisational behaviour. It helps design tasks that not only improve user performance but also enhance user motivation with the job.

In terms of the *crowd*, there are many theories about the motivation related to the relationships or interaction among users. A dominant motivation of people in participating in a group is to compare themselves with others and then improve themselves based on this comparison (explained in social comparison theory). Two other theories, common identity theory and common bond theory, describe the attachments of users as members of a group. They can attach to the group as a whole (common identity) or to some individuals of the group (common bond). Besides that, legitimate peripheral participation (LPP) is a theory about how to move from initial to experienced members in the community. Another aspect related to the crowd in collaborative working environments is individual effort in the group. The collective effort model (CEM) provides a framework to explain this issue.

Finally, in terms of the *requester*, in many cases, they have to consider using tangible rewards to encourage their users. The principal-agent theory is an efficient tool to design these kinds of incentives. However, providing rewards can diminish intrinsic motivations of recipients, hence motivation crowding theory can be used to deal with this. The following sections present more details of the above-mentioned theories.

Collective Effort Model (CEM)

CEM is a “model of individual effort on collective tasks”. This theory can be used to explain the phenomenon that people tend to make less effort when working collectively than working individually¹ (Karau and Williams, 1993).

¹This phenomenon is called *social loafing* or *free riding*.

Common Bond Theory, Common Identity Theory

These two theories are about attachment of a member in a group. They can be used to predict the causes and consequences of members' attachment to the group as a whole (common identity) or to group members individually (common bond) (Prentice et al., 2006; Ren et al., 2007).

Feedback Intervention Theory (FIT)

This theory concerns the effects of feedback intervention. According to FIT, after receiving feedback, people can direct their attention towards the task itself, other irrelevant tasks, or the recipients themselves. As a result of this, recipients can be motivated to stay with the system (if the attention is themselves) or to perform other tasks (if the attention is other irrelevant tasks), or they can understand more about the task that is related to the feedback (if the attention is the task itself). This theory can be used to maximise the effectiveness of feedback on performance results and user motivations (Kluger and DeNisi, 1996).

Flow Theory

Flow theory studies the causes of *flow*, a mental status where people are completely immersed in an activity and the activity itself is the reward regardless of its outcomes. Being in this status, people are intrinsically motivated by the activity (Nakamura and Csikszentmihalyi, 2002). This theory can be used to identify factors that can boost intrinsic motivation of users in crowdsourcing projects when undertaking activities in the systems.

Goal-Setting Theory

This is a theory which can be used to develop action plans that motivate and guide people towards their goals. The premise of this theory is that conscious goals affect action. In order to have motivational goals, the following five factors are needed to consider: goal specificity, goal difficulty, goal commitment, feedback, and task complexity (Locke et al., 1981; Locke and Latham, 2002).

Information Gap Theory

This theory postulates that people will be curious when there is a gap between the information that they already know and the information they want to know. This theory can be used to stimulate the curiosity of users in a crowdsourcing project while they are performing tasks or participating in other activities, hence retaining them longer with the project (Law et al., 2016) or encouraging them come back to the project on a regular basis.

Principal-Agent Model

The principal-agent model analyses the problem of how to set payments when an employer or a manager (called *principal*) hires an employee (called *agent*) to perform tasks. In many cases, the principal cannot monitor actions of the agent strictly and also cannot infer the agent's action based on the outcomes of the tasks. In these cases, the principal can affect the agent's action by providing incentives (in the form of a contract) based on observable signals that are correlated with the action (Laffont and Martimort, 2002). This model can be used to design appropriate incentives to improve user performance in crowdsourcing systems.

Job Characteristic Model (JCM)

JCM is a model that posits the effects of five core job characteristics (i.e., skill variety, task identity, task significance, autonomy and direct feedback from the job) on work outcomes (work motivation, job satisfaction, work effectiveness, etc.) through three critical psychological states (i.e., experienced meaningfulness of the work, experienced responsibility for outcomes of the work and knowledge of the actual results of the work activities) (Hackman and Oldham, 1980). In this thesis, JCM is used to identify task-related design objectives that affect user motivations.

Legitimate Peripheral Participation (LPP)

LPP is a theory that seeks to understand the move from newcomers to established members and then core members in a community of practice or collaboration (Lave and Wenger, 1991; Jackson et al., 2015). This theory can be used in choosing appropriate approaches to support the move from initial to sustained participation.

Motivation Crowding Theory

Motivation crowding studies the effects of external interventions on intrinsic motivation. Rewards diminish (or enhance) intrinsic motivation when people are perceived to be controlling (or supporting) (Frey and Jegen, 2001). This theory can be used when considering to deliver rewards or punishments in order to have optimal effects on user motivation.

Self-Determination Theory

SDT is a theory of motivation. SDT identifies three innate psychological needs that are considered to exist in every person: *competence* (the need to master and understand the environment, for example, how to perform tasks), *relatedness* (the need for a sense of attachment and belonging to other people), and *autonomy* (the need to control their own behaviours). The theory differentiates two types of motivation: *extrinsic motivation* (motivated by external rewards such as money, recognition or power) and *intrinsic motivation* (motivated by the task itself such as interest or enjoyment) (Ryan and Deci, 2000). This theory can be used in understanding user motivation in a crowdsourcing project or encouraging user participation by moving gradually from extrinsic motivations to intrinsic ones, which is called *internalisation*.

Social Comparison Theory

This theory explains the tendency of people to compare themselves with individuals around them. According to Festinger (1954), humans have a drive to evaluate their opinions and abilities. In many cases, people's tendency is to compare with others who have some similarities in order to evaluate themselves, and then they are motivated to improve their abilities towards the level of performance of the ones they compare to². In the crowdsourcing context, each user takes part in activities that can interact with others in the community of the project or system. Thus, social comparison theory can be used to find proper approaches that can encourage participation and performance of users by providing an environment for them to compare with others and continually improve themselves.

²In social comparison theory, this phenomenon is called *unidirectional drive upward*.

Bibliography

- Shipra Agrawal and Navin Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. page 26, 2012.
- Eman Aldhahri, Vivek Shandilya, and Sajjan Shiva. Towards an effective crowdsourcing recommendation system: A survey of the state-of-the-art. In *Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium On*, pages 372–377. IEEE, March 2015. ISBN 978-1-4799-8356-8.
- Maha Alsayasneh, Sihem Amer-Yahia, Eric Gaussier, Vincent Leroy, Julien Pilourdault, Ria Mae Borromeo, Motomichi Toyama, and Jean-Michel Renders. Personalized and diverse task composition in crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 30(1):128–141, January 2018. ISSN 1041-4347.
- Venkatachalam Anantharam, Pravin Varaiya, and Jean Walrand. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-Part I: I.I.D. rewards. *IEEE Transactions on Automatic Control*, 32(11):968–976, November 1987. ISSN 0018-9286.
- Ricardo Matsumura Araujo. 99designs: An analysis of creative competition in crowd-sourced design. In *Proceedings of the 1st AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, pages 17–24. AAAI Press, November 2013.
- Nikolay Archak and Arun Sundararajan. Optimal design of crowdsourcing contests. In *Proceedings of the 13th International Conference on Information Systems (ICIS)*, pages 1–16. AIS, December 2009.
- Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1):31–45, February 2014. ISSN 0364-765X, 1526-5471.
- Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002a.
- Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, January 2002b. ISSN 0097-5397, 1095-7111.

- Segal Avi, Gal Ya'akov, Ece Kamar, Eric Horvitz, Alex Bowyer, and Grant Miller. Intervention strategies for increasing engagement in crowdsourcing: Platform, predictions, and experiments. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3861–3867. AAAI Press, July 2016.
- Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. *Journal of the ACM*, 65(3):1–55, March 2018. ISSN 00045411.
- Kinjal Basu and Souvik Ghosh. Analysis of Thompson sampling for Gaussian process optimization in the bandit setting. *arXiv:1705.06808 [stat]*, February 2018.
- James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, February 2012.
- Donald A. Berry, Robert W. Chen, Alan Zame, David C. Heath, and Larry A. Shepp. Bandit problems with infinitely many arms. *The Annals of Statistics*, 25(5):2103–2116, October 1997. ISSN 0090-5364.
- Arpita Biswas, Shweta Jain, Debmalya Mandal, and Y. Narahari. A truthful budget feasible multi-armed bandit mechanism for crowdsourcing time critical tasks. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1101–1109. IFAAMAS, 2015.
- Patrick Brandtner, Andreas Auinger, and Markus Helfert. Principles of human computer interaction in crowdsourcing to foster motivation in the context of open innovation. In *HCI in Business*, pages 585–596. Springer, 2014.
- Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv:1012.2599 [cs]*, December 2010.
- Sébastien Bubeck, Gilles Stoltz, Csaba Szepesvári, and Rémi Munos. X-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, May 2011.
- Roberto Calandra, Andre Seyfarth, Jan Peters, and Marc Peter Deisenroth. An experimental comparison of Bayesian optimization for bipedal locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1951–1958. IEEE, May 2014. ISBN 978-1-4799-3685-4.
- Chris Callison-Burch. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1, pages 286–295. ACL, August 2009.
- Judy Cameron, Katherine M. Banko, and W. David Pierce. Pervasive negative effects of rewards on intrinsic motivation: The myth continues. *The Behavior Analyst*, 24(1):1–44, 2001.

- Ruggiero Cavallo and Shaili Jain. Efficient crowdsourcing contests. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 2, pages 677–686. IFAAMAS, June 2012.
- Ruggiero Cavallo and Shaili Jain. Winner-take-all crowdsourcing contests with stochastic production. In *Proceedings of the 1st AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, pages 34–41. AAAI Press, November 2013.
- Christopher P. Cerasoli, Jessica M. Nicklin, and Michael T. Ford. Intrinsic motivation and extrinsic incentives jointly predict performance: A 40-year meta-analysis. *Psychological Bulletin*, 140(4):980–1008, 2014. ISSN 1939-1455, 0033-2909.
- Dana Chandler and Adam Kapelner. Breaking monotony with meaning: Motivation in crowdsourcing markets. *Journal of Economic Behavior & Organization*, 90:123–133, June 2013. ISSN 01672681.
- Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson sampling. In *Advances in Neural Information Processing Systems 24 (NIPS)*, pages 2249–2257. Curran Associates, Inc., 2011.
- Arghya Roy Chaudhuri and Shivaram Kalyanakrishnan. Quantile-regret minimisation in infinitely many-armed bandits. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 425–434. AUAI Press, August 2018.
- Shuchi Chawla, Jason D. Hartline, and Balasubramanian Sivan. Optimal crowdsourcing contests. In *Proceedings of the 23d Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 856–868. SIAM, January 2012.
- Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver, and Nando de Freitas. Bayesian optimization in AlphaGo. *arXiv:1812.06855 [cs, stat]*, December 2018.
- Vickie Curtis. Motivation to participate in an online citizen science game a study of FoldIt. *Science Communication*, 37(6):723–746, 2015.
- George E. Dahl, Tara N. Sainath, and Geoffrey E. Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8609–8613. IEEE, May 2013. ISBN 978-1-4799-0356-6.
- Peng Dai, Jeffrey M. Rzeszotarski, Praveen Paritosh, and Ed H. Chi. And now for something completely different: Improving crowdsourcing workflows with micro-diversions. In *CSCW '15 Proceedings of the 2015 ACM Conference on Computer Supported Cooperative Work*, pages 628–638. ACM, 2015. ISBN 978-1-4503-2922-4.
- Edward L. Deci. Effects of externally mediated rewards on intrinsic motivation. *Journal of Personality and Social Psychology*, 18(1):105, 1971.

- Edward L. Deci, Richard Koestner, and Richard M. Ryan. A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. *Psychological Bulletin*, 125(6):627–668, 1999. ISSN 1939-1455, 0033-2909.
- Dominic DiPalantino and Milan Vojnovic. Crowdsourcing and all-pay auctions. In *EC '09 Proceedings of the 10th ACM Conference on Electronic Commerce*, pages 119–128. ACM, July 2009.
- Anhai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96, April 2011. ISSN 00010782.
- Hossein Esfandiari, Amin Karbasi, Abbas Mehrabian, and Vahab Mirrokni. Batched multi-armed bandits with optimal regret. *arXiv:1910.04959 [cs]*, October 2019.
- Leon Festinger. A theory of social comparison processes. *Human Relations*, 7(2):117–140, January 1954. ISSN 0018-7267.
- Oluwaseyi Feyisetan and Elena Simperl. Beyond monetary incentives: Experiments in paid microtask contests modelled as continuous-time markov chains. *arXiv:1901.05670 [cs]*, January 2019.
- Peter I. Frazier. A tutorial on Bayesian optimization. *arXiv:1807.02811 [cs, math, stat]*, page 20, July 2018.
- Bruno S. Frey and Reto Jegen. Motivation crowding theory. *Journal of Economic Surveys*, 15(5):589–611, December 2001. ISSN 0950-0804, 1467-6419.
- Bruno Giovanni Galuzzi, Ilaria Giordani, A. Candelieri, Riccardo Perego, and Francesco Archetti. Bayesian optimization for recommender system. In Hoai An Le Thi, Hoai Minh Le, and Tao Pham Dinh, editors, *Optimization of Complex Systems: Theory, Models, Algorithms and Applications*, volume 991, pages 751–760. Springer International Publishing, 2020. ISBN 978-3-030-21802-7 978-3-030-21803-4.
- Zijun Gao, Yanjun Han, Zhimei Ren, and Zhengqing Zhou. Batched multi-armed bandits problem. In *Advances in Neural Information Processing Systems 32 (NIPS)*, page 11, 2019.
- David Geiger and Martin Schader. Personalized task recommendation in crowdsourcing information systems—current state of the art. *Decision Support Systems*, 65:3–16, September 2014. ISSN 01679236.
- Antonio Ghezzi, Donata Gabelloni, Antonella Martini, and Angelo Natalicchio. Crowdsourcing: A review and suggestions for future research. *International Journal of Management Reviews*, 20(2):343–363, April 2018. ISSN 14608545.
- Uri Gneezy and Aldo Rustichini. Pay enough or don't pay at all. *Quarterly Journal of Economics*, 115(3):791–810, August 2000.

- Jean-Bastien Grill, Michal Valko, Remi Munos, and Remi Munos. Black-box optimization of noisy functions with unknown smoothness. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 667–675. Curran Associates, Inc., 2015.
- J. Richard Hackman and Greg R. Oldham. *Work Redesign*. Organization Development. Addison-Wesley, 1980. ISBN 978-0-201-02779-2.
- Christopher Harris. 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)*, pages 15–18. ACM, February 2011.
- Jan Hartmann, Alistair Sutcliffe, and Antonella De Angeli. Investigating attractiveness in web user interfaces. In *CHI ’07 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 387–396. ACM, 2007.
- Jan Hartmann, Alistair Sutcliffe, and Antonella De Angeli. Towards a theory of user judgment of aesthetics and user interface quality. *ACM Transactions on Computer-Human Interaction*, 15(4):1–30, November 2008. ISSN 10730516.
- James Heyman and Dan Ariely. Effort for payment: A tale of two markets. *Psychological Science*, 15(11):787–793, 2004.
- Chien-Ju Ho, Aleksandrs Slivkins, and Jennifer Wortman Vaughan. Adaptive contract design for crowdsourcing markets: Bandit algorithms for repeated principal-agent problems. *Journal of Artificial Intelligence Research*, 55:317–359, February 2016. ISSN 1076-9757.
- Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963. ISSN 01621459.
- Andreas Holzinger. Usability engineering methods for software developers. *Communications of the ACM*, 48(1):71–74, January 2005. ISSN 00010782.
- Jeff Howe. The rise of crowdsourcing. *Wired Magazine*, 14(6):1–4, 2006.
- Shih-Wen Huang and Wai-Tat Fu. Don’t hide in the crowd!: Increasing social transparency between peer workers improves crowdsourcing outcomes. In *CHI ’13 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 621–630. ACM, 2013.
- Akihiko Itoh and Shigeo Matsubara. Designing incentives for crowdsourced tasks via multi-armed bandits. In *IEEE International Conference on Agents (ICA)*, pages 70–73. IEEE, September 2016.

- Corey Brian Jackson, Carsten Osterlund, Gabriel Mugar, Katie DeVries Hassman, and Kevin Crowston. Motivations for sustained participation in crowdsourcing: Case studies of citizen science on the role of talk. In *HICSS '15 Proceeding of the 48th Hawaii International Conference on System Sciences*, pages 1624–1634. IEEE Computer Society, January 2015. ISBN 978-1-4799-7367-5.
- Shweta Jain, Ganesh Ghalme, Satyanath Bhat, Sujit Gujar, and Y. Narahari. 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)*, pages 86–94. IFAAMAS, May 2016.
- M.E. Johnson, L.M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2):131–148, October 1990. ISSN 03783758.
- Donald R Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.
- Kwang-Sung Jun, Kevin Jamieson, Robert Nowak, and Xiaojin Zhu. Top Arm Identification in Multi-Armed Bandits with Batch Arm Pulls. In *AISTATS '16 Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, page 14, 2016.
- Ece Kamar and Eric Horvitz. Incentives for truthful reporting in crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 3, pages 1329–1330. IFAAMAS, June 2012.
- Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised Bayesian optimisation via Thompson sampling. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 84, page 10, 2018.
- Yunus Emre Kara, Gaye Genc, Oya Aran, and Lale Akarun. Actively estimating crowd annotation consensus. *Journal of Artificial Intelligence Research*, 61:363–405, February 2018. ISSN 1076-9757.
- Steven J. Karau and Kipling D. Williams. Social loafing: A meta-analytic review and theoretical integration. *Journal of Personality and Social Psychology*, 65(4):681–706, 1993.
- Nicolas Kaufmann, Thimo Schulze, and Daniel Veit. 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*, volume 11, pages 1–11. AIS, August 2011.

- Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *STOC '08 Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 681–690. ACM Press, May 2008. ISBN 978-1-60558-047-0.
- Avraham N. Kluger and Angelo DeNisi. The effects of feedback interventions on performance: A historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psychological Bulletin*, 119(2):254–284, 1996. ISSN 1939-1455, 0033-2909.
- Robert E. Kraut, Paul Resnick, Sara Kiesler, Moira Burke, Yan Chen, Niki Kittur, Joseph Konstan, Yuqing Ren, and John Riedl. *Building Successful Online Communities: Evidence-Based Social Design*. MIT, 2011. ISBN 978-0-262-01657-5.
- Chinmay Kulkarni, Koh Pang Wei, Huy Le, Daniel Chia, Kathryn Papadopoulos, Justin Cheng, Daphne Koller, and Scott R. Klemmer. Peer and self assessment in massive online classes. In *Design Thinking Research*, pages 131–168. Springer, 2015.
- H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97, 1964. ISSN 00219223.
- Jean-Jacques Laffont and David Martimort. *The Theory of Incentives: The Principal-Agent Model*. Princeton University, 2002. ISBN 978-0-691-09183-9 978-0-691-09184-6.
- T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, March 1985. ISSN 01968858.
- Jean Lave and Etienne Wenger. *Situated Learning: Legitimate Peripheral Participation*. Learning in Doing. Cambridge University, 1991. ISBN 978-0-521-41308-4 978-0-521-42374-8.
- Edith Law, Ming Yin, Joslin Goh, Kevin Chen, Michael A. Terry, and Krzysztof Z. Gajos. Curiosity killed the cat, but makes crowdwork better. In *CHI '16 Proceedings of the 2016 Conference on Human Factors in Computing Systems*, pages 4098–4110. ACM, 2016. ISBN 978-1-4503-3362-7.
- Dan Li and Evangelos Kanoulas. Bayesian optimization for optimizing retrieval systems. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 360–368. ACM Press, 2018. ISBN 978-1-4503-5581-0.
- Haifang Li and Yingce Xia. Infinitely many-armed bandits with budget constraints. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 2182–2188. AAAI Press, February 2017.
- Kimberly Ling, Gerard Beenen, Pamela Ludford, Xiaoqing Wang, Klarissa Chang, Xin Li, Dan Cosley, Dan Frankowski, Loren Terveen, Al Mamunur Rashid, Paul Resnick, and Robert Kraut. Using social psychology to motivate contributions to online communities. *Journal of Computer-Mediated Communication*, 10(4):00–00, July 2005. ISSN 10836101.

- Edwin A. Locke and Gary P. Latham. Building a practically useful theory of goal setting and task motivation: A 35-year odyssey. *American Psychologist*, 57(9):705–717, 2002. ISSN 0003-066X.
- Edwin A. Locke, Karyll N. Shaw, Lise M. Saari, and Gary P. Latham. Goal setting and task performance: 1969-1980. *Psychological Bulletin*, 90(1):125–152, 1981. ISSN 0033-2909.
- Alex Luedtke, Emilie Kaufmann, and Antoine Chambaz. Asymptotically optimal algorithms for budgeted multiple play bandits. *Machine Learning*, 108(11):1919–1949, November 2019. ISSN 0885-6125, 1573-0565.
- Tie Luo, Sajal K. Das, Hwee Pink Tan, and Lirong Xia. Incentive mechanism design for crowdsourcing: An all-pay auction approach. *ACM Transactions on Intelligent Systems and Technology*, 7(3):1–26, February 2016. ISSN 21576904.
- Tie Luo, Salil S. Kanhere, Hwee-Pink Tan, Fan Wu, and Hongyi Wu. Crowdsourcing with tullock contests: A new perspective. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 2515–2523. IEEE, April 2015.
- Wenlong Lyu, Pan Xue, Fan Yang, Changhao Yan, Zhiliang Hong, Xuan Zeng, and Dian Zhou. An efficient Bayesian optimization approach for automated optimization of analog circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(6):1954–1967, June 2018. ISSN 1549-8328, 1558-0806.
- T Malone. Toward a theory of intrinsically motivating instruction. *Cognitive Science*, 5(4):333–369, December 1981. ISSN 03640213.
- Thomas W. Malone. Heuristics for designing enjoyable user interfaces: Lessons from computer games. In *CHI '82 Proceedings of the Conference on Human Factors in Computing Systems*, pages 63–68. ACM, 1982.
- Winter Mason and Duncan J. Watts. Financial incentives and the “performance of crowds”. *ACM SigKDD Explorations Newsletter*, 11(2):100–108, 2010.
- David Issa Mattos, Erling Mårtensson, Jan Bosch, and Helena Holmström Olsson. Optimization experiments in the continuous space: The limited growth optimistic optimization algorithm. In Thelma Elita Colanzi and Phil McMinn, editors, *Proceedings of the 10th International Symposium on Search-Based Software Engineering (SSBSE)*, volume 11036, pages 293–308. Springer International Publishing, 2018. ISBN 978-3-319-99240-2 978-3-319-99241-9.
- Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.
- Benny Moldovanu and Aner Sela. The optimal allocation of prizes in contests. *American Economic Review*, 91(3):542–558, January 2001.

- Jeanne Nakamura and Mihaly Csikszentmihalyi. The concept of flow. *Handbook of Positive Psychology*, pages 89–105, 2002.
- Cuong Nguyen, Onook Oh, Abdulrahman Alothaim, Triparna de Vreede, and Gert Jan de Vreede. Engaging with online crowd: A flow theory approach. In Lakshmi S. Iyer and Daniel J. Power, editors, *Reshaping Society through Analytics, Collaboration, and Decision Support*, volume 18, pages 175–189. Springer, 2015. ISBN 978-3-319-11574-0 978-3-319-11575-7.
- Quoc Viet Hung Nguyen, Thanh Tam Nguyen, Lam Ngoc Tran, and Karl Aberer. An evaluation of aggregation techniques in crowdsourcing. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Xuemin Lin, Yannis Manolopoulos, Divesh Srivastava, and Guangyan Huang, editors, *Web Information Systems Engineering – WISE 2013*, volume 8181, pages 1–15. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-41153-3 978-3-642-41154-0.
- Alexandru Niculescu-Mizil. Multi-armed bandits with betting. In *COLT’09 Workshop on On-Line Learning with Limited Feedback*, page 4, 2009.
- Oghenekarho Okobiah, Saraju Mohanty, and Elias Kougiannos. Fast design optimization through simple kriging metamodeling: A sense amplifier case study. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(4):932–937, April 2014. ISSN 1063-8210, 1557-9999.
- Vianney Perchet, Philippe Rigollet, Sylvain Chassang, and Erik Snowberg. Batched bandit problems. *The Annals of Statistics*, 44(2):660–681, April 2016. ISSN 0090-5364.
- Christine Phillips and Barbara S. Chaparro. Visual appeal vs. usability: Which one influences user perceptions of a website more? *Software Usability Research Lab*, 11(2):1–9, October 2009.
- Canice Prendergast. The provision of incentives in firms. *Journal of Economic Literature*, 37(1):7–63, March 1999.
- Deborah A. Prentice, Dale T. Miller, and Jenifer R. Lightdale. Asymmetries in attachments to groups and to their members: Distinguishing between common-identity and common-bond groups. In *Small Groups: Key Readings*, Key Readings in Social Psychology, pages 83–95. Psychology Press, 2006. ISBN 978-0-86377-593-2 978-0-86377-594-9.
- Sarvapali D. Ramchurn, Trung Dong Huynh, Matteo Venanzi, and Bing Shi. Collabmap: Crowdsourcing maps for emergency planning. In *Proceedings of the 5th Annual ACM Web Science Conference (WebSci)*, pages 326–335. ACM, May 2013.

- Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006. ISBN 978-0-262-18253-9.
- Yuqing Ren, R. Kraut, and S. Kiesler. Applying common identity and bond theory to design of online communities. *Organization Studies*, 28(3):377–408, March 2007. ISSN 0170-8406.
- Herbert Robbins. Some aspects of the sequential design of experiments. 58(5):527–535, 1952.
- Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 321–328. AAAI Press, July 2011.
- Richard M. Ryan and Edward L. Deci. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, 55(1):68–78, January 2000.
- Ognjen Scekcic, Hong-Linh Truong, and Schahram Dustdar. Incentives and rewarding in social computing. *Communications of the ACM*, 56(6):72–82, June 2013. ISSN 00010782.
- Sandip Sen, Anton Ridgway, and Michael Ripley. 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)*, pages 137–144. IFAAMAS, May 2015.
- Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, January 2016. ISSN 0018-9219, 1558-2256.
- Aaron D. Shaw, John J. Horton, and Daniel L. Chen. Designing incentives for inexperienced human raters. In *CSCW '11 Proceedings of the 2011 ACM Conference on Computer Supported Cooperative Work*, pages 275–284. ACM, 2011.
- David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. ISSN 0028-0836, 1476-4687.
- Henri Simula. The rise and fall of crowdsourcing? In *Proceeding of the 46th Hawaii International Conference on System Sciences (HICSS)*, pages 2783–2791. IEEE, January 2013. ISBN 978-1-4673-5933-7 978-0-7695-4892-0.

- Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25 (NIPS)*, volume 2, pages 2951–2959. Curran Associates, Inc., 2012.
- Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, and Mostofa Ali Patwary. Scalable Bayesian optimization using deep neural networks. page 10, 2015.
- Rion Snow, Brendan O’Connor, Daniel Jurafsky, and Andrew Y. Ng. 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)*, pages 254–263. ACL, October 2008.
- Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, page 8, 2010.
- D. Steinkraus, I. Buck, and P.Y. Simard. Using GPUs for machine learning algorithms. In *Proceedings of the 2005 8th International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 1115–1120. IEEE, 2005. ISBN 978-0-7695-2420-7.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press, second edition edition, 2018. ISBN 978-0-262-03924-6.
- Shirin Tavara. Parallel computing of support vector machines: A survey. *ACM Computing Surveys*, 51(6):1–38, January 2019. ISSN 03600300.
- William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, December 1933. ISSN 00063444.
- Oksana Tokarchuk, Roberta Cuel, and Marco Zamarian. Analyzing crowd labor and designing incentives for humans in the loop. *IEEE Internet Computing*, 16(5):45–51, September 2012. ISSN 1089-7801.
- Long Tran-Thanh, Archie Chapman, Jose Enrique Munoz De Cote Flores Luna, Alex Rogers, and Nicholas R. Jennings. Epsilon-first policies for budget-limited multi-armed bandits. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1211–1216. AAAI Press, July 2010.
- Long Tran-Thanh, Archie C. Chapman, Alex Rogers, and Nicholas R. Jennings. Knapsack based optimal policies for budget-limited multi-armed bandits. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1134–1140. AAAI Press, July 2012.

- Long Tran-Thanh, Trung Dong Huynh, Avi Rosenfeld, Sarvapali D. Ramchurn, and Nicholas R. Jennings. 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)*, pages 477–484. IFAAMAS, May 2014.
- Francesco Trovo, Stefano Paladino, Marcello Restelli, and Nicola Gatti. Budgeted multi-armed bandit in continuous action space. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI)*, pages 560–568. IOS Press, September 2016. ISBN 978-1-61499-671-2.
- Nhat Van-Quoc Truong, Sebastian Stein, Long Tran-Thanh, and Nicholas R. Jennings. Incentive engineering framework for crowdsourcing systems. In *The 4th AAAI Conference on Human Computation and Crowdsourcing*, November 2016.
- Nhat Van-Quoc Truong, Sebastian Stein, Long Tran-Thanh, and Nicholas R. Jennings. Adaptive incentive selection for crowdsourcing contests. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 2100–2102. IFAAMAS, July 2018.
- Nhat Van-Quoc Truong, Sebastian Stein, Long Tran-Thanh, and Nicholas R. Jennings. 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)*, pages 85–97. Springer International Publishing, August 2019.
- Matteo Venanzi, John Guiver, Pushmeet Kohli, and Nicholas R. Jennings. Time-sensitive bayesian information aggregation for crowdsourcing systems. *Journal of Artificial Intelligence Research*, 56:517–545, July 2016. ISSN 1076-9757.
- Yizao Wang, Jean-yves Audibert, and Rémi Munos. Algorithms for infinitely many-armed bandits. In *Advances in Neural Information Processing Systems 21 (NIPS)*, page 8. Curran Associates, Inc., 2008.
- Yingce Xia, Tao Qin, Weidong Ma, Nenghai Yu, and Tie-Yan Liu. Budgeted multi-armed bandits with multiple plays. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, page 7. AAAI Press, 2016.
- Jiang Yang, Lada A. Adamic, and Mark S. Ackerman. Crowdsourcing and knowledge sharing: Strategic user behavior on Taskcn. In *Proceedings of the 9th ACM Conference on Electronic Commerce (EC)*, pages 246–255. ACM, July 2008.
- Ming Yin and Yiling Chen. Bonus or not? Learn to reward in crowdsourcing. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 201–207. AAAI Press, July 2015.

- Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. Taskrec: A task recommendation framework in crowdsourcing systems. *Neural Processing Letters*, 41(2):223–238, April 2015. ISSN 1370-4621, 1573-773X.
- Shuhan Zhang, Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, Xuan Zeng, and Xiangdong Hu. An efficient multi-fidelity Bayesian optimization approach for analog circuit synthesis. In *Proceedings of the 56th Annual Design Automation Conference 2019 on - DAC '19*, pages 1–6. ACM Press, 2019. ISBN 978-1-4503-6725-7.
- Haichao Zheng, Dahui Li, and Wenhua Hou. Task design, motivation, and participation in crowdsourcing contests. *International Journal of Electronic Commerce*, 15(4):57–88, July 2011. ISSN 1086-4415.
- Datong P Zhou and Claire J Tomlin. Budget-constrained multi-armed bandits with multiple plays. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4572–4579. AAAI Press, February 2018.
- Haiyi Zhu, Amy Zhang, Jiping He, Robert E. Kraut, and Aniket Kittur. Effects of peer feedback on contribution: A field experiment in Wikipedia. In *CHI '13 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2253–2262. ACM, 2013.
- Gabe Zichermann and Christopher Cunningham. *Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps*. O'Reilly, 1st edition, 2011. ISBN 978-1-4493-9767-8.
- Matthew Zook, Mark Graham, Taylor Shelton, and Sean Gorman. Volunteered geographic information and crowdsourcing disaster relief: A case study of the Haitian earthquake. *World Medical & Health Policy*, 2(2):6–32, January 2010. ISSN 1948-4682.