



Thompson sampling-based recursive block elimination for dynamic assignment under limited budget in pure-exploration

Shameem Puthiya Parambath¹ · Christos Anagnostopoulos¹ · Saleh Abdullah M. Alfahad¹

Received: 7 December 2023 / Accepted: 1 December 2024 / Published online: 18 December 2024
© The Author(s) 2024

Abstract

In this paper, we investigate Thompson sampling-based sequential block elimination approaches for dynamic assignment problems in a pure-exploration Multi-Armed Bandit (MAB) setting with limited budget constraints. The problem can be considered as a bandit game-play between the environment and a decision-maker in a metric space. Many instances of problems in fields such as e-commerce, logistics, mobility management, data management and operations research can be framed as dynamic assignment problems with budget constraints. Given an l -dimensional action space representing l variants of an entity and a budget for exploring the action space, the optimal dynamic assignment problem refers to the task of identifying the values to be assigned to different variants of the entity that maximizes the total reward by utilizing at most the given budget of rounds of play. We contribute a class of block elimination-based MAB algorithms specifically designed for the dynamic assignment problem with limited budget. Our algorithms begin by discretizing the continuous action space into a finite set of discrete actions, then proceed with a recursive block elimination procedure to remove sub-optimal actions. The elimination is carried out by calculating confidence bounds over blocks of actions. We explore two different confidence bound estimation techniques. We perform comprehensive experiments on two problem instances from distributed data management and logistics. Our results showcase that our approach yields a lower misidentification probability (i.e., the probability of recommending a

Responsible editor: Eyke Hüllermeier

✉ Shameem Puthiya Parambath
sham.puthiya@glasgow.ac.uk

Christos Anagnostopoulos
christos.anagnostopoulos@glasgow.ac.uk

Saleh Abdullah M. Alfahad
2426473a@student.gla.ac.uk

¹ School of Computing Science, University of Glasgow, 18 Lilybank Gardens, Glasgow, Scotland G12 8RZ, UK

non-optimal action) compared to state-of-the-art elimination-based pure-exploration MAB algorithms.

Keywords Sequential learning · Multi-armed bandits · Thompson sampling · Learning with feedback · Pure-exploration · Dynamic assignment · Dynamic pricing

1 Introduction

It is very common among online business providers to market different variants of an entity. Here, the term ‘entity’ is used in a very generic sense, referring to any application-specific object such as a service, product, or even a machine learning model. Instances of such online business practices can be observed in various domains, including logistics, mobility management, e-commerce, and big-data management systems, among many others. In logistics, it is very common for users to choose from different delivery services for the same source-destination pair. For example, DHL provides express and economy services, while UPS offers standard and express services. Mobility management service providers like Bolt, Lyft, etc., allow customers to select from different variants of ride services, such as standard or shared. Similarly, in e-commerce, it is common to find different variants of the same product available for purchase, such as a pack of five or ten chocolate bars. Many standard problems that arise in data management systems can also be viewed as instances of this broader problem. In Federated Learning (FL), pruning techniques are used to compress global predictive models for different compression (pruning) ratios. These compressed models are then distributed among edge nodes (e.g., end-user devices) (Jiang et al. 2022) for different prediction tasks. These compressed models can be considered as variants of the global model. Another closely related problem is finding the optimal hyperparameters of a machine learning model during training phase, as introduced in Jamieson and Talwalkar (2016).

A recurring problem across these aforementioned application domains is the challenge of finding the optimal values to be assigned to different services/products to maximize expected rewards or revenues. In logistics operations, including mobility management and e-commerce, this problem is akin to determining the optimal prices for different variants of services (e.g., standard vs. express) to maximize revenue. In the case of FL pruning problem, the optimal pruning ratio varies for different nodes due to the heterogeneity of the edge nodes and network infrastructure. The objective here is to find pruning ratios that minimize communication and computational costs (Jiang et al. 2022). The primary challenge in these problem settings is that the demand curve for services and products or the data distribution at the networked nodes is not known to the decision-maker (algorithm) in advance. Nonetheless, the decision-maker has the flexibility of continuously changing the assigned values (e.g., pruning ratios or prices) and often randomizing those changes to ‘learn’ the optimal values. Historically, multi-armed bandit (MAB) algorithms have been employed to learn optimal assignment values adaptively. In practical settings like logistics and e-commerce, price experiments are conducted using the principle of “learn, then earn” to

estimate optimal prices. In contrast, MAB algorithms provide the flexibility of “learn-while-earn” (Misra et al. 2019), which enables faster data-driven decision-making and enhances the overall experimental design. MAB-based adaptive learning algorithms have been proposed for dynamic pricing (Misra et al. 2019; Mueller et al. 2019; Luo et al. 2021; Mussi et al. 2022), as well as for determining optimal pruning ratios in federated learning (Jiang et al. 2022).

A Multi-armed bandit (MAB) is a feedback-driven sequential learning framework that models the trade-off between *exploration* and *exploitation* over a sequence of actions (Lattimore and Szepesvári (2020); Cesa-Bianchi and Lugosi 2006). In the MAB framework, the decision-maker sequentially explores the set of actions to find the action with the optimal reward. In the past, MAB-based algorithms have been employed in web-based news recommendation systems (Li et al. 2010) and online advertising (McMahan and Streeter 2009). In the classical stochastic reward MAB model, the focus is on the exploration-exploitation trade-off. This involves exploring the action space to estimate rewards and exploiting the most rewarding actions from the explored set to minimize cumulative regret. Here, we specifically consider the problem of optimal dynamic assignments in pure-exploration settings. Unlike in classical settings, in the pure-exploration setting, there is no explicit trade-off between exploration and exploitation. In the pure-exploration setting we consider, the goal is to find the best-performing action within a given fixed but **limited** budget T . That is, after exploring the actions for a relatively small number of rounds, often much smaller than the number of actions k , the decision-maker recommends an action that it considers the best one. Importantly, this recommended action need not be the same as the action played at the end of the exploration. Next, we argue that the pure-exploration setting aligns closely with the problem settings we discussed above.

It has been established that online prices do not change frequently and exhibit relatively long spells of fixed prices (Gorodnichenko et al. 2018). Moreover, the majority of online users are price-sensitive (Hult et al. 2019; Breugelmans and Campo 2016); consequently, evaluating revenues is costly due to the high sensitivity of users to price changes: lower prices may result in losses, while higher prices could lead to non-purchases. The same cost considerations apply in big-data management settings, where computational and communication costs impact the performance of downstream application systems. Evaluating the computational and communication costs for *every* possible pruning ratio combination is infeasible and may result in operational failures (Jiang et al. 2022). Therefore, to avoid such shortcomings, a strategy that explores the action space for a fixed but limited budget, while estimating optimal values such as pruning ratios or prices, is deemed appropriate. In this context, instead of framing the dynamic assignment as an exploration-exploitation trade-off, one can explore the actions exhaustively within the constraints of the given budget to adaptively learn the demand curve or data distribution and then exploit the most promising action based on the acquired knowledge. In the problems discussed above, the budget is limited, and refers to the number of users exposed to a particular price or the number of pruning ratio combinations attempted. These quantities are directly linked to T , the number of trials (rounds) in the MAB framework, as the number of users or pruning ratio combinations can be adjusted by setting T . In summary, MAB in pure-exploration provides the

advantage of the “learn-while-earn” paradigm, allowing the identification of optimal pruning/price regions dynamically while respecting the budget constraints.

The three main characteristics of the problems we study here are:

- The action space is infinite, as actions correspond to assignment values spanning a given range (e.g., pruning ratios in the range $(0,1)$)
- The reward shows structural properties, based on the observation that similar actions result in similar rewards. This is assumed in various problems, including continuous Multi-Armed Bandits (MABs) (Bubeck et al. 2011).
- The budget is limited, meaning the number of rounds is much less than the number of actions ($T < k$). This constraint is crucial as the number of actions is prohibitively large to try every action at least once.

It is already established that classical Multi-Armed Bandit (MAB) algorithms may perform poorly when applied in pure-exploration settings (Bubeck et al. 2011). Pure-exploration algorithms have been historically employed for best-arm identification (Audibert and Bubeck 2010; Bubeck et al. 2011; Gabillon et al. 2012; Shahrampour et al. 2017) and outlier detection problems (Zhuang et al. 2017; Ban and He 2020; Zhu et al. 2020). However, applying such algorithms directly to dynamic assignment problems with a limited budget is challenging, as they require testing *all* action combinations *at least* once. Moreover, these algorithms assume a finite and relatively small action space (Audibert and Bubeck 2010; Shahrampour et al. 2017; Zhuang et al. 2017; Ban and He 2020; Zhu et al. 2020). Most infinite or continuous-armed bandit algorithms leverage the structural properties of the action space (e.g., Lipschitz property) and discretize the action space to derive efficient algorithms (Magureanu et al. 2014; Podimata and Slivkins 2021). Unfortunately, these algorithms, often based on the popular Upper Confidence Bound (UCB) algorithm (Auer et al. 2002), also require sampling each discretized action at least once. Given the impracticality of evaluating revenues or estimating network costs by enumerating all combinations, and considering the high cost associated with these evaluations, sampling all actions even once becomes unfeasible.

A possible workaround is to use coarse-grained discretization to limit the number of actions. However, such an approach might result in the optimal action or nearly optimal action to be excluded from the discretized action set. An orthogonal approach relies on *action elimination*, where potentially sub-optimal actions are removed during exploration based on an estimated confidence value (Karnin et al. 2013; Shahrampour et al. 2017; Tirinzoni and Degenne 2022). To the best of our knowledge, all the elimination-based Multi-Armed Bandit (MAB) algorithms proposed in the literature so far either require sampling each action at least once (Karnin et al. 2013; Tirinzoni and Degenne 2022) or involve sampling only a representative action, such as the most densely connected one (Shahrampour et al. 2017). In this paper, we contribute a novel *block elimination* MAB algorithm. Our approach is distinct from previous elimination-based pure-exploration algorithms, as the decision-maker recursively eliminates *blocks of actions* by estimating a confidence score over the blocks. Our algorithms are based on Thompson Sampling, a simple yet competitive heuristic for Multi-Armed bandit problems with a provable optimality guarantee (Chapelle and Li 2011; Agrawal and Goyal 2012, 2013). Traditionally, Thompson Sampling has been used in classical

settings, with the top-two sampling version of Thompson Sampling often employed in pure-exploration settings (Russo 2016; Jourdan et al. 2022). In addition to the vanilla Thompson Sampling-based schemes, we also propose top-two variants of the block elimination strategy we develop. Unlike previous approaches, our mechanism does not necessitate sampling all actions before deciding whether an action should be eliminated or not. This enhances the feasibility and applicability of our algorithm in dynamic assignment problems with limited budgets. Additionally, all the pure-exploration Thompson Sampling based schemes are designed for fixed-confidence pure-exploration problem (ref Sect. 2) (Russo 2016; Jourdan et al. 2022; Wang and Zhu 2022), and their empirical analysis has been limited to binary rewards. We extend Thompson Sampling for fixed-budget pure-exploration with non-binary rewards.

The remainder of this paper is organized as follows: Sect. 2 provides an overview of related work and discusses our contributions. This section covers past work on infinite or continuous Multi-Armed Bandits (MABs), structured bandits, pure-exploration MAB algorithms, and MAB algorithms for finding optimal pruning ratios and dynamic pricing. Section 3 elaborates on the proposed recursive block elimination algorithm, discussing different confidence score estimation techniques. We also introduce the top-two variants of our proposed algorithm. Section 4 reports on our experimental results based on two different dynamic assignment problems: dynamic pricing in logistics and dynamic pruning in FL. We also conduct a sensitivity analysis of the proposed algorithms. Finally, Sect. 5 concludes the paper.

2 Related work

In this section, we discuss studies related to different aspects of the problem and the application settings we consider in this paper.

Structured MAB with infinite arms Many algorithms for continuous-armed (infinite arms) bandits have been proposed in the past. Multi-Armed Bandit (MAB) problems become challenging when the arm set is very large or infinite. However, in such scenarios, the expected rewards often exhibit structural properties that we can exploit when designing efficient algorithms. Many such algorithms assume that the reward function is Lipschitz continuous. In Magureanu et al. (2014), the authors proposed a Lipschitz bandit algorithm that discretizes the continuous space and uses a UCB-based algorithm. However, this approach is not feasible in our case, as discretization may still result in many arms compared to the budget. Another variant of the proposed algorithm makes use of adaptive discretization, gradually zooming in on the more promising regions of the action space. Unfortunately, that algorithm also requires sampling every action at least once. In Kleinberg et al. (2008), continuous-armed bandits in a one-dimensional metric space were considered, whereas (Kleinberg et al. 2019) considered continuous-armed bandits in generic metric spaces, assuming that the mean-payoff function is a Lipschitz function with respect to some dissimilarity measure defined in the space. The proposed “zooming” algorithm achieves essentially the best possible regret bound in a minimax sense with respect to the environments studied. Inspired by the zooming algorithm, Podimata and Slivkins (2021) adopted it to non-stochastic bandit settings where the rewards are assumed to be generated by

an oblivious adversarial environment. Algorithms based on tree partitions were also proposed for problems with infinite arms. Bubeck et al. (2011) proposed a tree-based optimization algorithm that recursively partitions the space. While this work is similar to ours, there are significant differences. The above works do not clearly mention how the action space is discretized to sample the actions. Moreover, these works consider bandits in the exploration-exploitation setting rather than the pure-exploration setting. Most crucially, these algorithms don't eliminate suboptimal blocks from future exploration.

Pure-exploration algorithms In the last decade, numerous algorithms for pure-exploration bandits have been proposed. The majority of these algorithms are based on the popular UCB algorithm (Auer et al. 2002). The pure-exploration bandit problem is studied from two standpoints: (i) fixed-confidence settings and (ii) fixed-budget settings.

In the fixed-confidence setting, the objective is to minimize the number of rounds to find the action with the given probability confidence with respect to the best action. In the fixed-budget setting, the goal is to find the best action using, at most, the given number of rounds. Fixed-confidence algorithms optimize the budget for a given confidence level, whereas fixed-budget algorithms focus on minimizing the number of trials needed to achieve a fixed confidence level. The general consensus is that the fixed-budget setting problem is significantly harder (Carpentier and Locatelli 2016).

Since our work is in the fixed budget setting, we will discuss literature pertaining to fixed-budget pure-exploration bandit algorithms only. Interested readers can refer to Tirinzoni and Degenne (2022); Zhuang et al. (2017); Audibert and Bubeck (2010); Carpentier and Locatelli (2016); Gabillon et al. (2012) and the references therein for a detailed overview of fixed-confidence pure-exploration algorithms.

It has already been shown that elimination-based algorithms are better suited in fixed-budget settings as they can find the optimal actions provably (Carpentier and Locatelli 2016). The crux of any elimination-based algorithm is to iteratively eliminate suboptimal actions from the action set until a unique action or near-optimal action is left.

In the classical paper (Audibert and Bubeck 2010), the authors proposed the successive reject algorithm. The algorithm divides the trial budget into $k - 1$ elimination phases and successively removes the arm with the lowest empirical mean at the end of each phase. One of the main drawbacks of this algorithm is that it requires every action to be sampled at least once and thus cannot be used in a situation where the action set is very large compared to the budget. Another main drawback is that the successive rejection algorithm requires prior knowledge of a problem-dependent parameter.

Sequential Halving (SH) (Karnin et al. 2013) algorithm is similar to the successive reject algorithm. Like in the successive reject case, in SH, the given budget is split evenly across $\log_2 T$ elimination phases where T is the budget. Within each round, actions are sampled uniformly to update the empirical mean values. At the end of the elimination round, the worst-performing half of the actions, in terms of empirical mean, are eliminated. Like in the successive reject algorithm, SH requires every action to be sampled at least once and thus cannot be used when the action set is larger than the budget.

In Shahrampour et al. (2017), authors propose a block elimination-based algorithm named Sequential Block Elimination (SBE). The algorithm also splits the budget evenly across the elimination phase, and in each elimination phase, instead of a single action, a block of action is removed. The algorithm is based on using the side information of the actions. The side information is utilized by forming action blocks and sampling the central action in each round. This is based on the assumption that the central arm is the most connected action in each block. At the end of each elimination phase, the worst-performing block, in terms of the empirical mean reward of the central action, is removed. The important thing to keep in mind is that this algorithm uses block elimination, but it plays only the central action from each block and makes the elimination decision based on the mean reward of this action.

Recently, the authors in Faella et al. (2020) proposed the variance-based rejects (VBR) algorithm. The algorithm is similar to the successive reject and sequential halving algorithm. The algorithm proceeds in phases, and in each phase, all the non-eliminated actions are sampled for a fixed number of rounds to estimate the corresponding empirical mean reward and standard deviation. The main difference between VBR and other algorithms is that the elimination criterion is based on a confidence score on the empirical mean rewards, not solely on the empirical mean reward. All the actions whose upper bound is lower than the current maximal lower bound are eliminated after each elimination phase. This algorithm also cannot be scaled into the problem settings where the action set is much larger than the budget.

Bandits for dynamic assignment problems The majority of the previous work in the dynamic assignment is restricted to the dynamic pricing problem due to its application in e-commerce. In dynamic pricing, a company is trying to automatically optimize the prices of some services/products to maximize revenue. The user arrives sequentially, and the decision-maker sets the price. The user will only purchase the product if the price is lower than their intrinsic valuation. The decision-maker will never observe the true valuation of the product, but only the revenue from the set price.

Thompson sampling is a simple yet efficient heuristic based on randomly sampling actions according to its success probability of being optimal. Recent studies showed that Thompson sampling can significantly outperform UCB-style algorithms in the classical MAB settings (Chapelle and Li 2011; Agrawal and Goyal 2013). In the past, top-two variants of the Thompson sampling algorithm were proposed for dynamic allocation in pure-exploration settings (Russo 2016; Jourdan et al. 2022). These algorithms are not elimination based, and at each round, the algorithm has to choose an action from the entire action set, irrespective of the suboptimality of many actions.

In Misra et al. (2019), authors proposed a UCB-inspired dynamic pricing algorithm. Like in the UCB algorithm, the proposed algorithm starts by sampling all the price points to construct partial demand information for different prices. This demand is used to select the next price point to sample. Since this algorithm is a UCB-style algorithm, it can not be used in our settings due to the reasons outlined earlier. Mueller et al. (2019) proposed a linear contextual bandit algorithm. The work considers dynamic pricing when the number of products is too large to be handled by standard bandit algorithms. They consider slightly different settings where the product demands are assumed to follow a linear model where the time-varying dynamics of the price-demand relation are captured using a positive semi-definite matrix. The setting is very

different from what we consider in this paper. The algorithm is specifically designed for the exploration-exploitation trade-off with the assumption that the underlying model evolves over time. This algorithm also requires all price points to be sampled at least once. Recently, many contextual dynamic pricing algorithms have been proposed. Xu and Wang (2022) proposed an extension to the famous EXP4 algorithm (Auer et al. 2002). In Luo et al. (2022), authors extended the UCB algorithm to consider context information. The algorithm assumes the user's valuation to be a linear function of the contexts and carries out exploration to estimate the parameters to employ UCB-style price recommendations. Both these approaches make use of context information and are designed for the case when the action set is smaller compared to the budget constraint. In this work, we consider the settings when the budget is very limited, and the number of actions is large.

Contributions: Our main technical contributions are:

- We propose a recursive block elimination method that utilizes Thompson Sampling to address dynamic assignment problems. This approach incorporates a confidence score based elimination strategy to improve the decision-making accuracy. Furthermore, we explore an extension of this method by integrating a top-two sampling strategy. This extension aims to further improve the efficiency by leveraging additional probabilistic insights.
- Our proposed method extends Thompson Sampling to fixed-budget pure-exploration, in contrast to the more common fixed-confidence pure-exploration framework (ref Sect. 2). Additionally, departing from conventional binary reward settings used in the Thompson Sampling, we extend Thompson Sampling with non-binary reward values.
- We introduce two distinct techniques for estimating confidence scores, drawing inspiration from popular confidence estimation schemes widely used in MAB literature. These methods adapt classical approaches to provide a more precise measurement of uncertainty in each round of decision making, catering specifically to the dynamic and stochastic nature of the environments.
- Through comprehensive experiments conducted on two distinct problem settings (dynamic pricing and dynamic pruning), we substantiate the superior performance of our algorithm compared to state-of-the-art algorithms found in the literature.

3 Thompson sampling based recursive block elimination framework

3.1 Preliminaries

Notations Throughout the paper, we use bold lowercase letters to denote vectors; for example, $\boldsymbol{\mu}$ represents the mean vector. The individual elements within these vectors are indicated using indexed regular font. For instance, μ_i refers to the i^{th} element of the mean vector $\boldsymbol{\mu}$. We use calligraphic letters to represent sets, such as \mathcal{A} , representing the set of actions. Additionally, superscripts are employed to denote time rounds, allowing for temporal tracking of values. For example $\boldsymbol{\mu}^t$ and μ_i^t specifically refer to the values of $\boldsymbol{\mu}$ and its i^{th} element at the t^{th} round, respectively.

Background and definitions A stochastic Multi-Armed Bandit (MAB) problem is characterized by the number of actions k , the budget or number of rounds T , and k probability distributions ν_1, \dots, ν_k associated with the actions $\mathcal{A} = 1, \dots, k$. These distributions, representing the unknown reward probabilities, are undisclosed to the decision-maker. At each round $t = 1, \dots, T$, the decision-maker selects an action I_t from the action set \mathcal{A} and receives a reward z_{I_t} drawn from the probability distribution linked to action I_t . It should be noted that the received reward is independent of past actions and observations. The consideration here is in the context of the general pure-exploration setting, as introduced in Jamieson and Talwalkar (2016). In this setting, after each round t , the decision-maker commits to an action denoted as J_t , which minimizes the expected loss (maximizes expected reward) with maximal confidence. It is essential to note that, although the decision-maker observes the reward for the action I_t played in round t , the evaluation is solely based on the recommendation J_t , distinguishing it from the classical MAB setting.

In the pure-exploration setting, the metric known as *simple regret* is employed as the loss measure. Simple regret is calculated as the difference between the mean reward of the optimal action and the mean reward of the recommended action. Let μ_1, \dots, μ_k represent the means of the respective distributions ν_1, \dots, ν_k , where $i^* = \operatorname{argmax}_{i \in \mathcal{A}} \mu_i$ and $\mu^* = \mu_{i^*}$. The simple regret for the decision-maker at round t is defined as $r_t = \mu^* - \mu_{J_t}$. It's important to note that the simple regret is closely related to the misidentification probability, which signifies the probability that the recommendation is not optimal, denoted as $\mathbb{P}(J_t \neq i^*)$. An insightful observation from Audibert and Bubeck (2010) establishes that the misidentification probability serves as an upper bound for the expected simple regret, i.e., $\mathbb{E}[r_t] \leq \mathbb{P}(J_t \neq i^*)$. This relationship underscores the significance of the misidentification probability in understanding the performance of the decision-maker in the pure-exploration context.

In the fixed budget pure-exploration setting, the decision-maker operates within a given budget of T rounds. The primary objective is to maximize the probability of correctly identifying the optimal action within these T rounds, effectively minimizing the misidentification probability. Notably, in situations characterized by a limited budget, the number of available actions (k) is often substantially larger than the allocated budget, i.e., $k \gg T$. This discrepancy is particularly evident in our specific applications, where the number of pruning ratios and price combinations far exceeds the available budgeted rounds (T). The ultimate goal is to determine the action (such as pruning ratio or price) that maximizes the reward while adhering to the constraints of the fixed budget of T rounds. In the context of dynamic pruning, the reward is gauged in terms of computational and communication gains, while in dynamic pricing, the focus shifts to revenue optimization.

We assume smooth reward functions, specifically Lipschitz continuous functions, following the approach in Magureanu et al. (2014); Kleinberg et al. (2019); Podimata and Slivkins (2021). Accordingly, our problem can be framed as a Lipschitz bandit problem within an l -dimensional metric space, where the metric is induced by the Lipschitz condition (Kleinberg et al. 2019). In this context, \mathcal{X} represents the l -dimensional metric space of actions, characterized by an unknown reward function satisfying the Lipschitz condition. In each round of the bandit process, the decision-maker selects an action (point) denoted as $x \in \mathcal{X}$ and receives a reward from the associated reward

distribution linked to x . The overarching objective is to identify the action with the highest expected reward. Importantly, in the application settings under consideration, the parameter l corresponds to the number of variants or distinct features of the products or services.

Remark 1 We acknowledge that assuming Lipschitz continuity may seem like a strong condition. However, it is important to note that, to our limited knowledge, all works on continuous-armed bandits depend on some form of smoothness and boundedness assumptions. It should also be noted that Lipschitz continuity is a weaker assumption compared to continuous differentiability or convexity. Specifically, in our cases, if the clusters are predefined, local Lipschitz continuity should be sufficient.

Thompson Sampling is a Bayesian heuristic for solving multi-armed bandit problems. At each time step, the decision-maker employs Thompson Sampling by sampling the parameters of the reward distribution from a specified prior distribution. The chosen action is then determined based on the posterior probability of being the optimal action. In the specific scenario of Bernoulli-distributed binary rewards, Thompson Sampling has demonstrated superior performance compared to UCB-based algorithms, even when there is a mismatch with the prior (it's noteworthy that the conjugate prior of the Bernoulli distribution is the Beta distribution). In our case, given that the rewards are positive real numbers, we assume that the rewards for each action are sampled from a log-normal distribution. To specify the priors for the mean and precision (inverse of the variance) of the underlying normal distribution, from which the log-normal distribution is derived, we use the following formulations:

At round t , precision (τ_i^t , inverse of variance) and mean (μ_i^t) of the action i is sampled from Gamma and Normal priors, respectively, as follows:

$$\begin{aligned} \tau_i^t | z_{I_1} \dots z_{I_{t-1}} &\sim \Gamma \left(\alpha + \frac{n_i^t}{2}, \beta + \frac{1}{2} \text{SSE}(i) + \frac{n_i^t (\bar{y}_i - \xi)^2}{2(n_i^t + 1)} \right) \\ \mu_i^t | \tau_i^t, z_{I_1} \dots z_{I_{t-1}} &\sim \mathcal{N} \left(\frac{n_i^t \bar{y}_i + 1}{n_i^t + 1}, (n_i^t + 1) \tau_i^t \right) \end{aligned} \quad (1)$$

In the expressions provided in Eq. (1), α , β , and ξ are fixed constants. The variable n_i^t represents the number of times the action i is selected up to round t . Additionally, $y_{I_t} = \ln(z_{I_t})$, where z_{I_t} denotes the reward associated with the action played at round t , I_t . The term \bar{y}_i corresponds to the empirical mean of y_{I_t} calculated over the rounds up to round t . The sum of squared errors for an action i , denoted as SSE, is defined as the summation of the squared differences between y_i and the empirical mean \bar{y}_i . Mathematically, this is expressed as follows:

$$\text{SSE}(i) = \sum_{a=1}^t (y_i - \bar{y}_i)^2 \mathbb{I}[I_a == i]$$

Given the prior mean (μ^t) and precision (τ^t) vectors, the rewards are sampled according to:

$$\mathbf{z}^{t+1} | \mu^t, \tau^t \sim \text{LogNormal} \left(\mu^t, \sqrt{\frac{1}{\tau^t}} \right). \quad (2)$$

It should be noted that we use z_i, \bar{z}_i to indicate the actual observed reward and the sampled reward from the posterior distribution for action i , respectively.

3.2 Block elimination mechanism

Our pure-exploration Multi-Armed Bandit (MAB) algorithm is based on the sequential elimination of actions. The algorithm begins by discretizing the continuous action space, a common practice in Lipschitz bandits to apply stochastic MAB algorithms like UCB (Podimata and Slivkins 2021; Magureanu et al. 2014). Previous methods for finding optimal pruning ratios (Jiang et al. 2022) and dynamic pricing (Misra et al. 2019) have also adopted discretized action sets. The granularity of the discretization is a key factor that controls the number of final actions to be sampled. Since our method does not require all actions to be sampled, we propose a fine-grained discretization. In our cases, we discretize all different ratio/price values independently into equal numbers so that they can be clustered (creating blocks of actions) in subsequent steps. As in other elimination-based approaches (Karnin et al. 2013; Shahrampour et al. 2017; Faella et al. 2020), our algorithm runs in phases. The pseudo-code for our proposed block-elimination approach is given in Algorithm 1. The algorithm takes as input the action space (\mathcal{X}), budget (T), the number of rounds in each phase (κ) and the number of clusters c . In each phase, we sample the actions for κ rounds to collect the rewards associated with those actions. After the end of every phase (κ rounds), the algorithm calculates the confidence scores and eliminates action blocks based on these scores. The exact details of how block confidence scores and rewards for the non-played actions are calculated are given in the following sections.

In parallel with block elimination, our method introduces a 'zooming' mechanism that recursively clusters actions, thereby decreasing the size of blocks (i.e., the number of actions in each block) after each phase. The zooming is achieved by further clustering all remaining blocks into c clusters after the elimination round. This approach streamlines the exploration to focus on the most promising actions among the non-eliminated actions. Finally, when the number of actions becomes too few to cluster, the algorithm shifts to eliminating individual actions based on confidence scores. This process continues until the budget is exhausted or only one action remains. After each round, our algorithm returns the action with the highest mean as the recommended choice.

The means and precisions of the rewards for all discretized actions are initialized using the optimistic initialization technique (ref Sect. 3.2.1) (Sutton and Barto 2018). In line:8, the decision-maker selects the action with the highest sampled reward based on the current prior estimations of the mean and precision of the rewards. Afterward, the actual reward from the chosen action is observed, and the mean reward and count

Algorithm 1: Recursive Block Elimination

Input : $\mathcal{X}, T, \kappa, c$

```

1  $\mathcal{A} = \text{Discretize}(\mathcal{X})$ ;
2  $\alpha = 1, \beta = 1, \xi = 1$ ;
3  $C = \text{cluster}(\mathcal{A}, c)$ ;
4 Initialize( $\mu^0$ ); // initialise mean reward vector
5 Initialize( $\tau^0$ ); // initialise precision vector
6 sample  $\mu^0$  &  $\tau^0$  according to Eq. (1);
7 for  $t = 1, 2, \dots, T$  do
8    $I_t = \text{argmax}_i \text{LogNormal}\left(\mu_i^{t-1}, \sqrt{\frac{1}{\tau_i^{t-1}}}\right)$ ;
9   play arm  $I_t$  and observe reward  $z_{I_t}$ ;
10  update mean reward and count for  $I_t$ ;
11  if  $(t\% \kappa == 0)$  then
12    Eliminate( $C, \mu^t, t$ )
13  sample  $\mu^t$  &  $\tau^t$  according to Eq. (1);
14 return action with highest mean;
```

variable for the chosen action are updated accordingly in line:10. The new mean reward and count variables are used to sample next prior values. After the κ -th round, the decision-maker invokes procedure `Eliminate` to eliminate actions or blocks of actions based on the calculated confidence scores and then zooms further.

The pseudo-code for `Eliminate` is provided in Algorithm:2. This procedure takes the discretized action set \mathcal{A} , the observed mean reward vector μ^t and the current round t as inputs. Initially, it checks whether the set of non-eliminated actions is large enough to be clustered or not. In each invocation of `Eliminate`, the algorithm reduces the block size, if possible, to zoom into the promising non-eliminated actions. To cluster actions, one can employ tree-based space-partitioning as in Coquelin and Munos (2007) or simple distance-based clustering algorithms like KMeans (Chawla and Gionis 2013). In line:2 of `Eliminate`, the decision-maker calculates the lower and upper confidence bounds on the mean reward for each cluster. The bound estimation procedure estimates the lower and upper bounds of each cluster when there are multiple clusters, and for each action when only one cluster is present. A visual demonstration and worked-out example of the proposed blocking algorithm are given at the end of this section.

Our elimination is based on the observation that the algorithm will not benefit by exploring actions whose potential mean rewards are lower compared to alternative actions with high rewards. Based on this observation, we devise an elimination rule: *we eliminate the blocks whose upper confidence bound on the mean reward value is lower than the maximum of the lower confidence bounds on the mean reward.*

3.2.1 Optimistic initialization

In bandit algorithms, an effective heuristic to foster exploration involves setting initial values that are optimistically high, rather than starting at zero or a minimal number. For instance, consider a scenario where the rewards are modeled as coming from a

Algorithm 2: Eliminate Process

```

1  $u = |C|$ ;
2  $(lbs, ubs) = \text{GetConfidenceBounds}(C, \mu)$ ;           // estimate lower and upper
   bounds
3  $maxlb = \max(lbs)$ ;
4 for  $\mathcal{S} \in C$  do
5   if  $(u == 1)$  then
6     if  $(|\mathcal{S}| == 1)$  then
7       return  $\mathcal{S}$ 
8     for  $i \in \mathcal{S}$  do
9       if  $(ubs[i] \leq maxlb)$  then
10        eliminate action  $i$  from  $\mathcal{S}$ 
11    return  $\mathcal{S}$ 
12 if  $(ubs[\mathcal{S}] \leq maxlb)$  then
13    $C = C \setminus \mathcal{S}$ ;           // eliminate action block  $\mathcal{S}$  from  $C$ 
14 return  $\text{cluster}(C)$ 

```

zero-mean Gaussian distribution with a unit standard deviation; here, an initial value of 0.5 is significantly optimistic. This approach of setting higher initial values proactively encourages the algorithm to sample actions that have not yet been explored. The rationale is that the initial high estimates make unexplored actions appear more attractive compared to those that have already been tested, which may have accumulated actual reward data that lowers their estimated values. To implement this strategy effectively, we initialize the mean and precision of the rewards with values that are tailored to the specifics of the problem at hand, ensuring that the exploration is both broad and aligned with the underlying reward distribution dynamics.

3.2.2 Updating rewards of non-observed actions and blocks

At each round, the decision-maker can only observe rewards from the action that was actually played. This limitation might lead to a biased sampling of actions since the mean and precision of actions that have never been played remain unchanged. To address this challenge, we begin by calculating the mean and precision for each block, based on the rewards of actions that have been played within that block. Subsequently, we adjust the mean and precision of the never-played actions within each block to match those of the block they are part of. For blocks where no actions have been played, we maintain the mean and precision at their initially set values, following the optimistic initialization technique described earlier. We ensure to update the mean and precision of all blocks at the end of each round to reflect the most recent data accurately. This method helps mitigate bias and promotes a more equitable representation of all actions in a block, irrespective of their play history.

Algorithm 3: Error-based Bound Estimation

Input : \mathcal{C}, μ, t
Initialize: γ

- 1 **for** $\mathcal{S} \in \mathcal{C}$ **do**
- 2 $\mu_{\mathcal{S}} = \text{meanreward}(c, \mu)$ // mean reward of block or action \mathcal{S}
- 3 $\sigma = \text{stderror}(\mathcal{S}, \mu)$ // standard error
- 4 $lbs_{\mathcal{S}} = \mu_{\mathcal{S}} - \gamma \cdot \sigma$; // lower confidence
- 5 $ubs_{\mathcal{S}} = \mu_{\mathcal{S}} + \gamma \cdot \sigma$; // upper confidence
- 6 **return** lbs, ubs

Algorithm 4: Hoeffding's Inequality-based Bound Estimation

Input : \mathcal{C}, μ, t

- 1 **for** $\mathcal{S} \in \mathcal{C}$ **do**
- 2 $\mu_{\mathcal{S}} = \text{meanreward}(\mathcal{S}, \mu)$ // mean reward of block \mathcal{S}
- 3 $\sigma = \sqrt{\frac{\log t}{2 \times N_{\mathcal{S}}}}$ // Hoeffding's Bound
- 4 $lbs_{\mathcal{S}} = \mu_{\mathcal{S}} - \sigma$; // lower confidence
- 5 $ubs_{\mathcal{S}} = \mu_{\mathcal{S}} + \sigma$; // upper confidence
- 6 **return** lbs, ubs

3.2.3 Confidence bound estimation

In Algorithm:2, we estimate the maximum lower bound at line:3 and subsequently evaluate the upper bounds for each block or action. This process leads to the removal of blocks or actions that yield the least reward. A critical challenge in this approach is that once a block or action is eliminated, it is not reconsidered; thus, each elimination decision must be made with a high degree of confidence. To address this, we implement two confidence estimation schemes, each designed to ensure that eliminations are both justified and irreversible, minimizing the risk of prematurely discarding potentially valuable actions or blocks. Additionally, in our implementation, we apply a strict threshold of five samples from each cluster before deciding on confidence estimation and elimination. This means that any block of actions with fewer than five samples is automatically excluded from consideration for elimination. This threshold ensures that our confidence estimates are based on a robust sample size, and every action block is considered with equal importance.

Standard error-based bound estimation: In Faella et al. (2020), the authors proposed a standard error-based elimination given in Algorithm:3. The γ is a constant that measures the confidence. We set $\gamma = 2$ in our implementation, achieving $\sim 90\%$ confidence interval.

Hoeffding's inequality-based bound estimation: This estimation technique is based on Hoeffding's inequality, which is the basis for the UCB algorithm. The pseudo-code is given in Algorithm:4, where $N_{\mathcal{S}}$ is the number of observed samples from the block $\mathcal{S} \in \mathcal{C}$.

3.2.4 Top-two sampling extension

In the Top-Two probability sampling policy, the decision-maker randomly selects either the action deemed most likely to be optimal or the second most likely optimal action, based on the current posterior probabilities. This approach introduces a structured randomness into the selection process, allowing for a balance between exploring less certain options and exploiting the actions believed to offer the highest potential reward. By alternating between these two top choices, the policy aims to refine the accuracy of the estimated rewards while mitigating the risk of local optima, thus enhancing the overall strategy for decision-making within stochastic environments. A top-two extension to Thompson sampling has been proposed in optimal allocation problem settings with strong theoretical guarantees (Russo 2016; Jourdan et al. 2022). We also consider the top-two version of our proposed algorithm given in Algorithm: 1. In the top-two version of our algorithm, we replace the original sampling scheme (line: 8) with Eqs. (3), (4) and (5). The rest of the algorithm remains the same.

$$J_1 = \arg \max_i \text{LogNormal} \left(\mu_i^{t-1}, \sqrt{\frac{1}{\tau_i^{t-1}}} \right) \quad (3)$$

$$J_2 = \arg \max_{i \neq J_1} \text{LogNormal} \left(\mu_i^{t-1}, \sqrt{\frac{1}{\tau_i^{t-1}}} \right) \quad (4)$$

$$I_t \sim \text{with probability } \theta \text{ sample from } \{J_1, J_2\} \quad (5)$$

In Eqs. (3) and (4), the algorithm finds the most likely optimal action and the second most likely action under the current posterior values, respectively. In Eq. (5), the algorithm randomly samples one of these actions, either the most likely optimal action or the second most likely option, with a probability of θ . The value of θ is often defaulted to 0.5, indicating equal probability for both actions.

3.2.5 Worked out example

To facilitate a better understanding of our algorithm, we provide a detailed illustrative example. In this scenario, we consider an action space represented in a two-dimensional plane as shown in Fig. 1. As outlined in the algorithm description, we begin by discretizing the action space. The discretized actions are denoted by \star symbols in Fig. 1. We assume four clusters and organize the actions into four blocks, represented as square blocks divided by bold white lines in the figure. We then sample 32 actions from the action space (for illustration we made the sampling evenly spanning across all clusters), selecting eight actions from each block. The played actions are highlighted in cyan (Block 1), green (Block 2), purple (Block 3), and red (Block 4) colors. The rewards associated with these 32 actions are detailed in Table 1.

For elimination purposes, we calculate the empirical mean and standard error for each block, as described in Sect. 3.2.2. The mean and standard error of a block are calculated as the mean and standard error of all the rewards in that block, respectively. Subsequently, we compute the lower and upper bounds (in this example, we

Table 1 Sampled revenues for the visual explanation of our algorithm

Blocks	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6	Sample 7	Sample 8
1	10.1	10.3	9.8	10.5	10.8	11	10.9	11.2
2	12.5	12.8	13.8	13	13.6	13.4	12.4	13.9
3	11.1	10.8	10.4	10.2	10	9.8	10.2	10
4	9.8	9.4	8.4	8.2	8.3	7.8	7.8	7.7

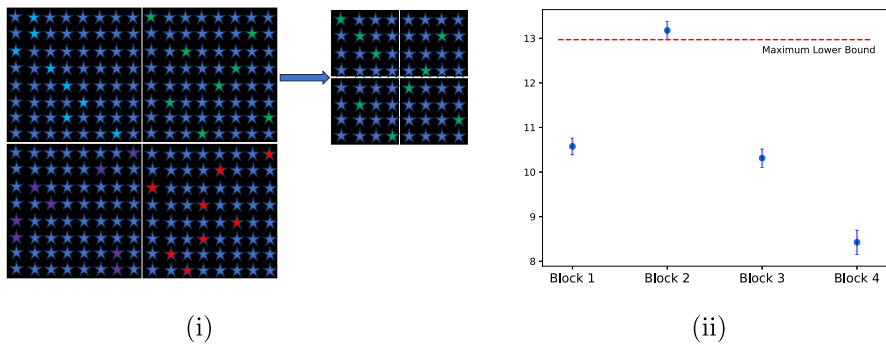


Fig. 1 Visual demonstration of the proposed algorithm. **i** initial blocks of actions and the remaining block after the elimination and **ii** visual representation of the upper and lower bounds of different blocks

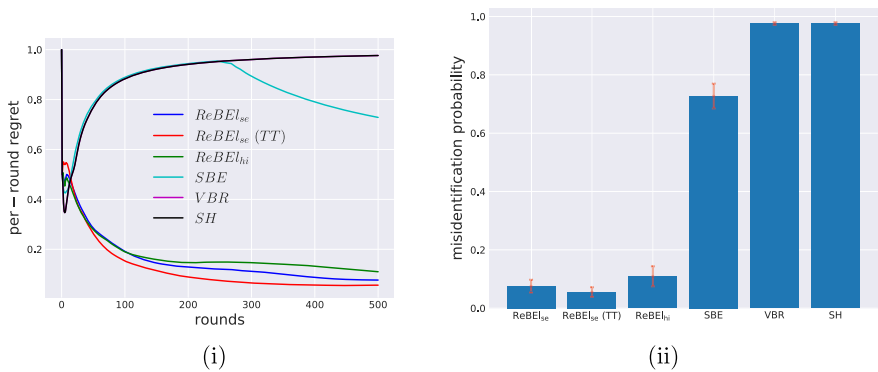


Fig. 2 Per-round regret and misidentification probability of different algorithms for the dynamic pricing problem

Table 2 Mean, Standard Error, Lower and Upper Bounds of the four blocks

Blocks	Mean	s.e	LB	UB
1	10.575	0.171	10.233	10.917
2	13.175	0.206	12.763	13.587
3	10.313	0.206	09.901	10.725
4	8.425	0.274	07.877	08.973

only consider standard error-based confidence score estimation, assuming $\gamma = 2$, as described in Sect. 3.2.3). The empirical mean, standard error, lower bound (LB), and upper bound (UB) of each block are provided in Table 2. This information is also visually represented in plot (ii) of Fig. 1. According to our elimination rule, we eliminate all blocks with an upper bound lower than the maximum lower bound. The red dotted line in the plot signifies the maximum lower bound, and all blocks whose maximum upper bound lies below this line will be eliminated. Since the upper bounds of all blocks are lower than the maximum lower bound of 12.763, corresponding to Block 2, we eliminate all blocks except Block 2.

We then repeat the same procedure with the non-eliminated blocks. These non-eliminated blocks of actions are further clustered into four clusters, and sampling is continued to estimate the mean rewards. This process is repeated until the end of the trials or until there are not enough actions available to cluster. The plots in Fig. 1 illustrates the further clustering of the non-eliminated blocks.

Remark 2 (Theoretical Analysis) We intend to delve into the theoretical analysis of our algorithm as part of our future work. This task necessitates the development of new tools that significantly differ from those used in existing elimination-based approaches. Addressing this analytical challenge will help us better understand the foundational principles of our algorithm and contribute novel insights to the field.

4 Experimental evaluation

In this section, we conduct a rigorous evaluation of our approach within real-world problem contexts. We focus on two dynamic assignment problems: (i) establishing optimal pricing strategies for parcel shipment services within logistics operations, and (ii) determining the most effective pruning ratios for Federated Learning (FL) applications. To accurately simulate these scenarios, we utilize synthetic datasets that are meticulously constructed based on real-world data. Our comprehensive evaluation involves a comparative analysis with leading elimination-based pure-exploration algorithms. Performance is assessed using metrics such as simple regret and the probability of misidentification. These synthetic datasets are intentionally designed to emulate the complexity and unique characteristics of the specific problem environments we are investigating.

In the logistics experiment, we consider the case $l = 2$, referring to two variants of a product/service. In the FL pruning experiment, we consider the case $l = 3$, referring to three variants of the product/service. In both cases, we used the K -means distance-based clustering (Chawla and Gionis 2013). The results strongly suggest that our block elimination algorithm is preferable to other sequential arm elimination algorithms in cases of a limited budget. We discuss the baselines and provide an in-depth analysis to verify the performance of our algorithm.

Note: The source code for the algorithms and data generation scripts can be found in the project page <https://github.com/shampp/tsrbe/>.

4.1 Performance metrics and baselines

Pure-exploration algorithms are compared using simple regret, while the misidentification probability is considered as an upper bound for the expected simple regret. The precise definitions of simple regret and misidentification probability can be found in Sect. 3.1. To gauge performance, we utilize *per-round* regret and misidentification probability as our key performance metrics. The per-round regret at round t is defined as the ratio of the cumulative simple regret to the round. Mathematically, at round t ,

it is given as follows:

$$\frac{1}{t} \sum_{j=1}^t r_j, \quad (6)$$

where r_j is the simple regret at round j . We compare our algorithm against three state-of-the-art elimination-based algorithms: (i) Sequential Halving (SH) (Karnin et al. 2013), (ii) Sequential Block Elimination (SBE) (Shahrampour et al. 2017), and (iii) Variance Based Rejects (VBR) (Faella et al. 2020).

Sequential halving (SH) Sequential Halving (Karnin et al. 2013) assumes a discrete action set. The algorithm operates by estimating the mean rewards for *all* actions in phases, with each phase comprising a fixed number of rounds. At the conclusion of each phase, the SH method incrementally eliminates actions from the set until only one remains. Within each phase, actions are uniformly sampled from the remaining non-eliminated actions to estimate empirical mean values. The elimination criterion at the end of each phase involves removing the worst half of the actions based on empirical mean values.

Sequential block elimination (SBE) Shahrampour et al. (2017) proposed SBE algorithm. The SBE algorithm operates in phases, wherein blocks of actions are eliminated at the conclusion of each phase. SBE initiates the process by creating blocks of actions and sampling the central action of each block in every round. This design is grounded in the assumption that the central action represents the most connected action within each block. After the elimination round, the block with the lowest empirical mean reward for its central action is removed. While both SBE and our algorithm are block elimination methods, a fundamental distinction lies in SBE's approach of sampling *only* the central action from each block and making elimination decisions based solely on the mean reward of this action.

Variance based rejects (VBR) The VBR algorithm (Faella et al. 2020) eliminates actions based on upper and lower bounds on the mean rewards, following a similar procedure to that of SH and SBE algorithms. At the conclusion of each phase, the algorithm estimates the bounds and eliminates all actions whose upper bound is lower than the maximal lower bound. In the VBR algorithm, actions are uniformly sampled within each phase, and the bound estimation scheme aligns with the approach outlined in Algorithm 3.

Our algorithm variant is named **ReBEL**, which stands for **R**ecursive **B**lock **E**limination. The Top-Two sampling variant of this algorithm is referred to as **ReBEL (TT)**. To distinguish between different confidence score estimation schemes, we use the specific estimation technique as a subscript. For instance, **ReBEL** utilizing the Standard Error-based confidence estimation technique is denoted as **ReBEL_{se}** and the version using Hoeffding's Inequality-based technique is referred to as **ReBEL_{hi}**. This nomenclature allows for clear differentiation and easy reference within discussions and analyses following below.

Remark 3 (Limited Budget Setting) In the existing literature, all pure-exploration algorithms mandate that each action be tried at least once. Consequently, for demonstration purposes, we display the results for $T > k$ in the initial experiment. This condition ensures that the total number of trials T exceeds the number of actions k , aligning with

the core requirements of pure-exploration methodologies and allowing us to effectively demonstrate the performance of our approach.

4.2 Case I: dynamic pricing for logistics operations

We initiated our evaluation by comparing the performance of our algorithm against baseline algorithms in a dynamic pricing scenario for logistics operations. The primary objective here was to determine the optimal prices for two variants of a logistic service to maximize total revenue. Specifically, we considered two types of logistic services ($l = 2$): (i) Standard and (ii) Express. For the Standard service, the minimum and maximum prices were set at 10 and 50, respectively. In contrast, the Express service prices ranged from 30 to 70 (in a specified monetary unit). We simulated the pricing dynamics of these ‘standard’ and ‘express’ shipping services using models based on real data from our industrial partner (Closed-Loop Data 2023).

To address the dynamic pricing problem, we discretized the 2-dimensional action space into 256 distinct actions and utilized K -Means clustering to create action clusters, thereby facilitating the block elimination process. The revenue, or reward, is modeled as a joint function derived from the proportion of customers purchasing services at given price pairs (standard, express). This revenue function is nonlinear, depending on the proportions of customers who opt for the services at specified price levels. Each price pair is transformed into probability scores using the standard logistic function. We then randomly sample the proportions of customers purchasing the corresponding services. To foster a positive initial bias and encourage exploration, we set the initial mean rewards $\mu^0 = 2.5$ and the initial precision to $\tau^0 = 2$. This optimistic initialization helps in initially guiding the exploration towards potentially underexplored yet profitable actions.

Revenue function The revenue function for the dynamic pricing experiment is modelled based on a recent study using real data (Puthiya Parambath et al. 2024; Sebastian Stein 2023) and is defined as:

$$r = p_1\lambda_1 + p_2\lambda_2, \quad (7)$$

where p_1 represents the probability of choosing the standard option, and p_2 is the probability of opting for the express service. Additionally, λ_1 and λ_2 denote the numbers of customers who chose options standard and express, respectively, based on the probabilities p_1 and p_2 . The probabilities are further defined as follows:

$$\begin{aligned} u_2 &= \frac{1}{1 + e^{(-\phi_2 a_2 - \psi_2)}} \\ u_1 &= (1 - u_2) \frac{1}{1 + e^{(-\phi_1 a_1 - \psi_1)}} \\ p_1 &= \frac{u_1}{u_1 + u_2} \quad \text{and} \quad p_2 = \frac{u_2}{u_1 + u_2} \end{aligned} \quad (8)$$

In the above $\phi_1 = -0.05$, $\phi_2 = -0.05$, $\psi_1 = -0.5$ and $\psi_2 = 2$ are constants and a_1 and a_2 are action pairs, i.e., prices for standard and express deliveries respectively.

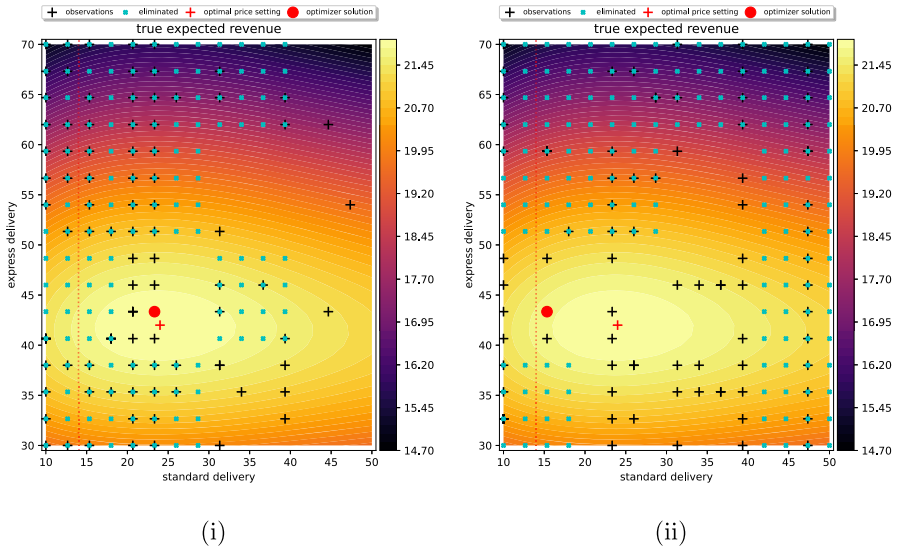


Fig. 3 Contour map of the action space and the running snapshot of **ReBEL_{se}** (i) and **ReBEL_{hi}** (ii) after the end of 100 rounds (Dynamic Pricing problem)

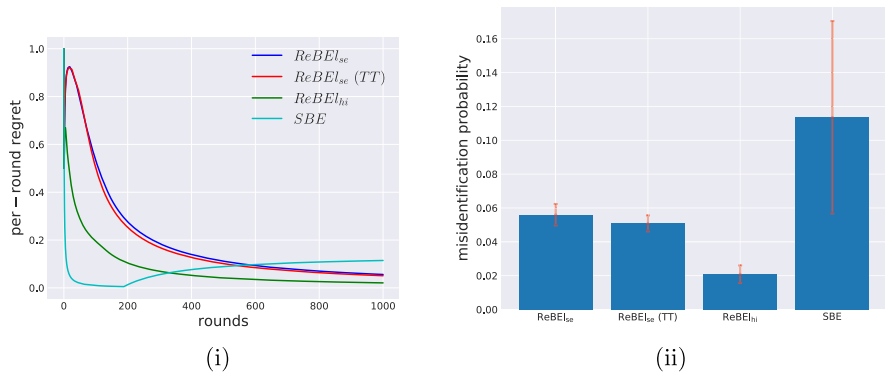


Fig. 4 Per-round regret and the misidentification probability of different block-elimination algorithms for the 3D dynamic pruning problem

4.2.1 Results and analysis

In Fig. 2, we present the results of our first experiments (Case I). Fig. 2i, ii illustrate the average per-round regret and the misidentification probability of all algorithms over 50 random executions with a budget $T = 500$, respectively. For this experiment, we set the cluster size (c) to 16, the number of actions $k = 256$, and the number of rounds for the elimination $\kappa = 1.5 \times c = 24$. Our strategies exhibit a lower misidentification probability compared to other methods. Specifically, **ReBEL_{se}**, **ReBEL_{se} TT**, and **ReBEL_{hi}** achieved the lowest misidentification probabilities and per-round simple regret. It is notable that among our proposed strategies, **ReBEL_{se}** performed marginally

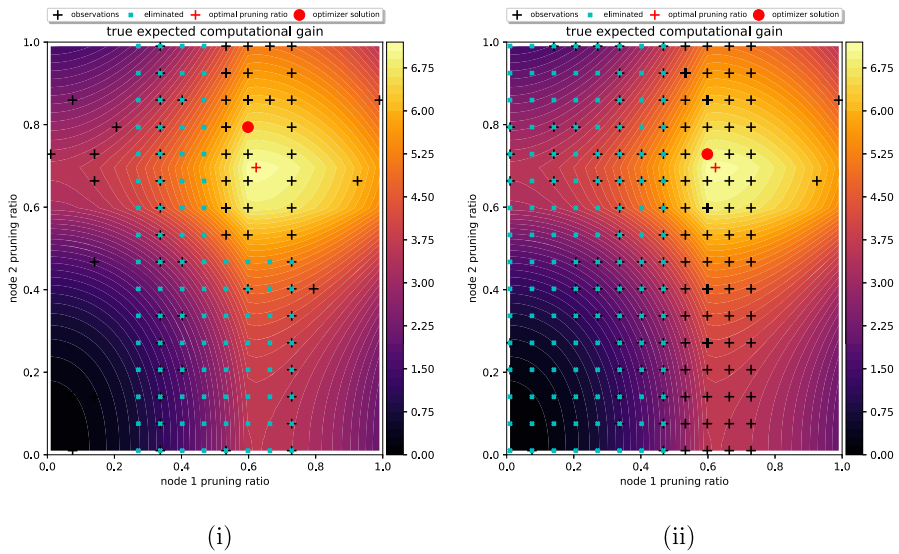


Fig. 5 Contour map of the action space, and the running snapshot of **ReBEL_{se}** (i) and **ReBEL_{hi}** (ii) after the end of 200 rounds (2D projection of the original dynamic pruning problem)

better than **ReBEL_{hi}**. This suggests that standard error provides a slightly more reliable confidence score than Hoeffding's inequality for this set of price distribution data. However, this is not universally the case, as will be demonstrated in our second set of experiments. As highlighted by theoretical studies such as those by Russo (2016) and Jourdan et al. (2022) Top-Two sampling achieved results that were marginally better than the conventional approach. Similar to the case of different confidence score estimation techniques, the significance of this result is not pronounced. This indicates a need for further experimentation to understand the nuances of these methods under varying conditions.

Among the block elimination strategies, it is clear that the performance of Sequential Block Elimination (SBE) does not match that of our proposed algorithms. The effectiveness of SBE is significantly influenced by the number of clusters and the geometry of the optimal region. In the dynamic pricing experiment, the performance of SBE notably benefits from a lower number of clusters, as demonstrated in the sensitivity analysis presented in Sect. 4.4. This improvement is primarily due to the substantial influence the geometry of the optimal region has on the overall performance of SBE. The optimal or near-optimal price region of logistic revenue distribution is relatively expansive, meaning that a smaller number of clusters ensures that optimal or near-optimal prices are close to the center of each cluster. Conversely, a larger number of clusters may lead to a fragmentation of this optimal region, resulting in degraded performance. This dynamic can be visualized and further explained using the contour plot shown in Fig. 3.

In this contour plot, the shipping price revenue function displays a smooth gradient, leading to only minor changes in revenue values around the optimal points, depicted by the innermost yellow ellipse. The color intensity exhibits minimal variation when

moving from the inner ellipse to the outer ellipses. In our experiment, we initially utilized SBE with 16 clusters, which resulted in considerable time spent estimating optimal rewards due to the wide dispersion of price points within each cluster. Once the optimal region was accurately identified, SBE began recommending optimal actions, as indicated by the decreasing per-round simple regret starting around round 300. However, our findings suggest that for our specific set of pricing data, SBE achieves better performance with fewer clusters (see Sect. 4.4). This configuration allows SBE to more swiftly locate the optimal prices, although it still underperforms compared to our newly proposed methods. This analysis underscores the need for careful cluster selection based on the characteristics of the pricing distribution to optimize the efficiency and effectiveness of the SBE strategy.

As anticipated, both SH (Successive Halving) and VBR (Value-Based Rejection) exhibited poor performance in our experiments. This outcome is largely due to the significant number of rounds these strategies devote to exploring sub-optimal arms before deciding to eliminate them. The results underscore that in scenarios with a limited budget, strategies that attempt to try all actions are sub-optimal. Instead, block elimination strategies have proven more effective. Among our three proposed block-elimination strategies- **ReBEL_{se}**, **ReBEL_{se} TT** and **ReBEL_{hi}**- the optimal action is identified within fewer than 100 rounds of the allocated budget. This efficiency contrasts sharply with that of SBE, which required close to 300 rounds, and other elimination strategies such as SH and VBR, which took over 500 rounds to locate the optimal action.

These findings highlight the efficiency of block elimination strategies in quickly converging to optimal actions, particularly in settings where resource constraints necessitate rapid decision-making. This comparative analysis clearly illustrates the advantages of the block elimination approach over more traditional methods, emphasizing its utility in optimizing performance within limited experimental budgets.

In addition, we present contours and snapshots of **ReBEL_{se}** and **ReBEL_{hi}** after the completion of a hundred rounds ($T = 100$) in Fig. 3. These visuals effectively illustrate how our algorithms manage to eliminate sub-optimal blocks of actions, regardless of the size and shape of the optimal region, in comparison to baseline strategies. Specifically, the plots showcase the *ground truth* optimal pair of solutions and the achieved optimal solution by **ReBEL_{se}** and **ReBEL_{hi}** after the conclusion of 100 rounds, i.e., the budget. Actions that are marked in cyan colour represent those that have been eliminated, while those marked with a '+' symbol indicate actions that have been played. Notably, both **ReBEL_{se}** and **ReBEL_{hi}** are successful in eliminating most suboptimal actions within the 100-round budget and consistently recommend actions that are closest to the optimal solution. Furthermore, these visuals provide insights into the performance of SBE. With only 16 clusters in this instance, the centroids often lie distant from the optimal region, contributing to the performance degradation of SBE. The sensitivity analysis further reveals that SBE performs better with a reduced number of clusters. The overall results underscore that, in scenarios with limited budgets where the budget is small compared to the number of actions ($T < k$), our algorithm consistently outperforms state-of-the-art algorithms. This demonstrates the robustness and efficiency of our block elimination strategies in optimizing the selection process under resource constraints.

4.3 Case II: dynamic pruning in federated learning

In our second experiment, we evaluated the performance of our algorithms against baseline methods in a three-dimensional problem setting focused on determining the optimal pruning ratios in Federated Learning (FL). FL techniques are instrumental in addressing privacy regulations and communication bandwidth limitations, facilitating distributed model training from decentralized data sources (McMahan et al. 2017). In FL, data are collected at edge nodes (e.g., smartphones) and used to update local models, which are then sent to a central server node for aggregation into a global model. This global model is subsequently shared among the edge nodes (McMahan et al. 2017). Given that typical deep learning models consist of millions of parameters, distributing, storing, or processing these models in their entirety at individual nodes is impractical. A prevalent approach in FL is to distribute a pruned version of the global model among the nodes to alleviate these challenges (Long et al. 2023). The heterogeneity of the nodes in FL-varying in computational power, storage capacity, and network connectivity - means that the optimal pruning ratio differs across nodes. Dynamic pruning, therefore, aims to tailor the pruning ratios to the specific needs and capabilities of each node to optimize performance and efficiency (Long et al. 2023; Jiang et al. 2022).

In our simulation of the Federated Learning (FL) scenario, we structured a network that includes a server node and three heterogeneous nodes connected to the server. With $l = 3$ (representing the three client nodes), our experimental actions are modeled in the three-dimensional space $[0, 1]^3$, reflecting a range of possible pruning ratios for each node. The computational and communication gains from each node in this network are quantified using a joint distribution model, where the gains for individual nodes are described by Poisson distributions based on the number of successful packets processed by the nodes. Each experimental round computes the mean of twenty-five samples, which correspond to taking twenty-five samples of the communication and computational gains for the same set of pruning ratios. This approach parallels the methodology used in our dynamic pricing experiment (Case I), where the parameter represented the number of customers. In the sensitivity analysis, we delve into how this parameter - the number of samples per round-affects the performance of our proposed algorithms. In this experiment, we aim to demonstrate that our algorithms can adaptively optimize the pruning ratios in real-time, leading to significant improvements in both communication efficiency and computational workload distribution across the nodes.

We discretized the action space into 4096 action triplets in the 3-dimensional plane and applied K -Means clustering to create 32 clusters. Considering the significant increase in the number of potential actions in this experiment compared to the dynamic pricing setup, we adapted our methodology to employ block elimination algorithms with a set budget of $T = 1000$. Although this budget is notably larger than that of the previous experiment, it is important to recognize that here the number of actions (k) far exceeds the budget (T), which is a scenario where $k \gg T$. As a result, non-block elimination based strategies such as SH and VBR were not included as baselines for this experiment. These methods require a proportionally higher budget

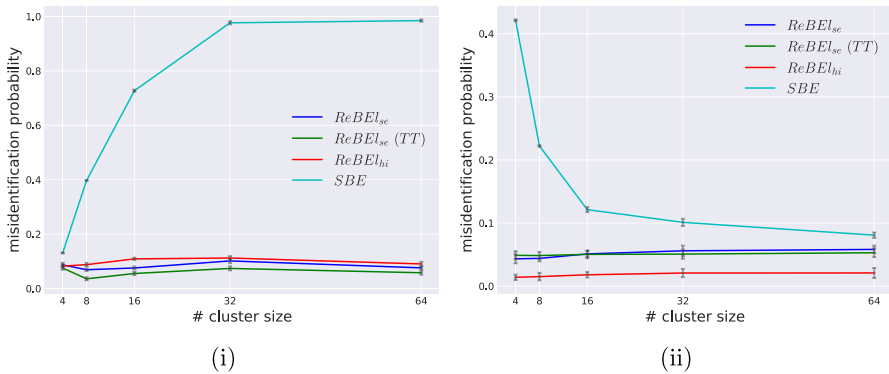


Fig. 6 Misidentification probability as a function of the number of clusters in **i** Dynamic Pricing and **ii** Dynamic Pruning experiments

relative to the number of actions to be effective, making them less suitable for the budgets constraints of this experiment. As in the dynamic pricing experiment, the results we present are based on averages over 50 random executions to ensure robustness and to minimize the impact of outliers or anomalous runs. To initiate the experiment under conditions that favor optimistic exploration, we set initial bias parameters with $\mu^0 = 1.3$ and $\tau^0 = 2$.

Reward function Like in the dynamic pricing experiment, we model the reward function based on a recent study (Puthiya Parambath et al. 2024; Sebastian Stein 2023). The reward function is defined as follows:

$$r = \frac{\lambda_1 p_1 + (1 - p_2)p_1 \lambda_2 + (1 - p_3)(1 - p_2)p_1 \lambda_3}{p_1 + (1 - p_2)p_1 + (1 - p_3)p_2} \quad (9)$$

In Eq. (9), λ_1 , λ_2 and λ_3 are the Poisson distributed random variable sampled according to the probabilities p_1 , p_2 and p_3 , respectively. The probabilities are defined as functions of pruning ratios:

$$\begin{aligned} p_1 &= 0.6 - |a_1 - 0.6| \\ p_2 &= |a_2 - 0.7| \\ p_3 &= |a_3 - 0.5| \end{aligned} \quad (10)$$

In Eq. (10), a_1 , a_2 and a_3 are the chosen action triplet, i.e., the pruning ratios for nodes 1, 2 and 3, respectively.

4.3.1 Results and analysis

The results of our pruning experiments in the context of FL are detailed and visualized in Fig. 4. In this figure: (i) illustrates the per-round regret, and (ii) depicts the misidentification probability. The different hyperparameters are set as follows: the number of actions at $k = 4096$, the number of clusters (c) at 32, and the number of rounds for elimination $\kappa = 1.5 \times c = 48$.

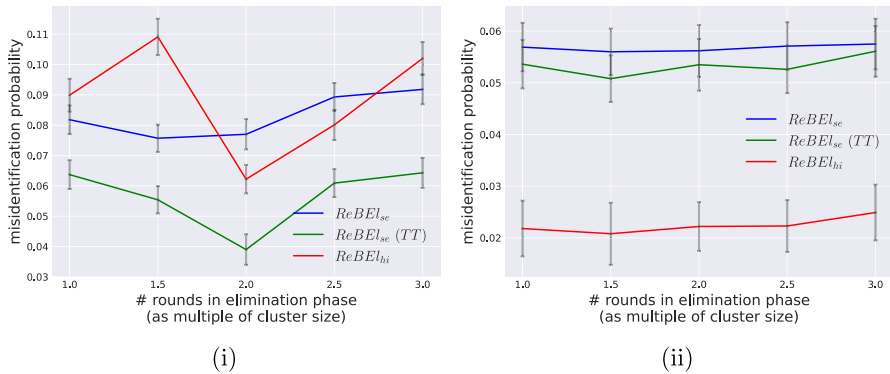


Fig. 7 Misidentification probability as a function of the number rounds to elimination phase in **i** Dynamic Pricing and **ii** Dynamic Pruning experiments

Observing the plots, it is evident that despite the relatively large action space compared to the budget (4096 against 1000), the block elimination strategies successfully eliminated suboptimal actions and frequently recommended optimal or near-optimal actions. Notably, in this experiment, the Hoeffding's inequality-based elimination strategy outperformed the standard error-based elimination strategy, though marginally. As in the case of dynamic pricing, the Top-Two sampling version demonstrated better performance than the vanilla version. The observed performance improvement, though marginal, with the Top-Two sampling version in both the dynamic pricing and the FL pruning experiments underscores a significant trend. This trend suggests that the Top-Two sampling strategy consistently outperforms any vanilla probabilistic sampling approaches in scenarios focused on pure exploration. This observation is in-line with the theoretical underpinnings.

It is noteworthy that **ReBEL_{se}**, **ReBEL_{se} TT**, and **ReBEL_{hi}** consistently resulted in lower per-round regret and misidentification probability compared to SBE. While the performance of SBE is better compared to the dynamic pricing experiments, particularly in the initial rounds, its superiority diminishes as the number of rounds increases. As depicted in the per-round regret plot, our proposed algorithms start yielding superior results after around 200 rounds.

The noticeable improvement in the performance of SBE in this experiment can indeed be attributed to the increased number of clusters and the specific characteristics of the optimal region. Our sensitivity study highlights that the effectiveness of SBE is significantly influenced by the number of clusters it utilizes. A larger number of clusters allows for finer segmentation of the action space, which can be particularly beneficial when the optimal region is narrow, as it increases the likelihood that one of the clusters is closely aligned with this optimal region. Additionally, the influence of the shape of the optimal region on SBE's performance is critical. As illustrated in the contour plot of the 2D projected action space in Fig. 5, the optimal action region in this experiment is relatively narrow, especially when compared to the broader, more dispersed optimal regions observed in the logistics experiment. In such settings, where the optimal region is narrow, having a larger number of clusters ensures that

the central action within each cluster is more likely to be near the optimal actions. This proximity can significantly enhance the ability of the algorithm to zero in on the most promising actions quickly. This configuration - larger number of clusters combined with a narrowly defined optimal region-helps explain why SBE performed notably better in this scenario. It effectively demonstrates how the structural aspects of the problem, such as the distribution and granularity of the action space, can directly impact the suitability and success of different baseline strategies.

As previously mentioned, SH or VBR were not engaged in this experimental setting, as these algorithms require at least the same number of rounds as the number of actions. Our focus here is specifically on cases where the budget is very limited. The results strongly reinforce the notion that in limited budget cases, where the budget is small compared to the number of actions ($T < k$), our algorithm consistently outperforms state-of-the-art algorithms.

In addition, we have included snapshots of **ReBEL_{se}** and **ReBEL_{hi}** after the completion of 200 rounds ($T = 200$) in the contour plot, of the 2D projection of the original problem, (see Fig. 5). Both algorithms successfully recommended the optimal action within the allocated budget of 200 rounds. The snapshots vividly illustrate the elimination strategies employed by each algorithm. **ReBEL_{hi}** efficiently eliminated the entire region of sub-optimal actions lying to the left of the optimal action, while **ReBEL_{se}** eliminated both the left and bottom halves.

4.4 Sensitivity analysis

In this section, we carry out the sensitivity analysis of the algorithms against two hyper-parameters of the proposed algorithms and the number of samples collected in each round. The two hyper-parameters of the algorithms are: the number of clusters s and the number of rounds to the elimination phase κ . The number of clusters impacts the performance of all block elimination-based algorithms: **ReBEL_{se}**, **ReBEL_{se} TT**, **ReBEL_{hi}** and, SBE, whereas the number of rounds to the elimination phase κ affects only **ReBEL_{se}**, **ReBEL_{se} TT** and **ReBEL_{hi}**.

In the initial sensitivity analysis, we examine the impact of varying the number of clusters on misidentification probability, upper bound to simple regret. This relationship is illustrated in Fig. 6, where Fig. 6i depicts the performance of the dynamic pricing experiment and Fig. 6ii depicts the performance of dynamic pruning experiment. In the context of logistic services, the performance of our proposed algorithms-**ReBEL_{se}**, **ReBEL_{se} TT**, and **ReBEL_{hi}**-demonstrates minimal variation with an increase in the number of clusters. It is noteworthy that for smaller cluster sizes, specifically when the number of clusters equals 4 and 8, there is a slight decrease in misidentification probability compared to scenarios with larger numbers of clusters. This increment is marginal but indicative of the sensitivity of the algorithm to cluster size at lower levels. Surprisingly, further increases in cluster size from 16 to 32 and 64 do not result in substantial improvements in misidentification probability. This trend can be attributed to the limited size of the action set, which suggests that the algorithms are capable of identifying the optimal action from a relatively small set of clusters. Consequently,

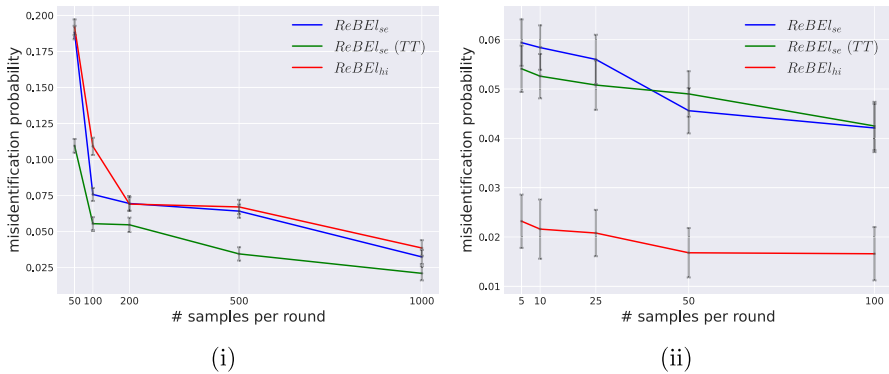


Fig. 8 Misidentification probability as a function of the number customers/trials sampled per round in **i** Dynamic Pricing and **ii** Dynamic Pruning experiments

there is no significant advantage in augmenting the number of clusters beyond a certain point. These findings are consistent with those observed in the original dynamic pricing experiments, suggesting that our algorithms are robust across different cluster configurations.

The misidentification probability plot for the SBE algorithm clearly indicates that its performance is significantly influenced by the number of clusters, even when the action set is small. Notably, with a smaller number of clusters (e.g., 4, as illustrated in the plot), the misidentification probability for SBE is on par with that of our proposed methods. However, as the number of clusters increases, the performance of SBE deteriorates markedly, with the misidentification probability approaching one. This outcome is primarily due to the geometric characteristics of the optimality region specific to this problem instance. In the scenario of dynamic pricing, where the optimality region tends to be smoother and larger, having a smaller number of clusters is advantageous. This smaller cluster configuration ensures that the centroid action, which represents the average position of the actions within a cluster, is likely to be closer to the optimal region. Consequently, a fewer number of clusters can be more effective for ensuring that SBE performs near optimally, as it reduces the complexity and variability within the action set, facilitating closer approximations to the optimal action.

A similar trend is evident in the dynamic pruning experiment, where the **ReBEL_{hi}** algorithm outperforms both **ReBEL_{se}** and **ReBEL_{se} TT**, even with a smaller number of clusters. A critical observation from the plot in Fig. 6 is that the performance of SBE aligns with that of the BESE algorithm as the number of clusters increases. This differs from the trend observed in the dynamic pricing results, where the performance of SBE deteriorated with more clusters. The key to understanding this variance lies in the geometry of the optimality region. In the dynamic pruning scenario, the optimality region is relatively small, and increasing the number of clusters enables SBE to better approximate the optimal action, enhancing its performance. Essentially, the effectiveness of the SBE algorithm is closely linked to the geometry of the optimality region and the number of clusters involved. In contrast, our proposed methods—**ReBEL_{hi}**, **ReBEL_{se}**, and **ReBEL_{se} TT**—exhibit a robust performance that is less influenced by these factors.

This resilience highlights their suitability for a broader range of scenarios where the shape and size of the optimality region may vary significantly.

In the second sensitivity analysis, we investigated the robustness of the algorithm against the number of rounds in the elimination phase, denoted as κ . We plotted the misidentification probability against κ in Fig. 7. Figure 7i depicts the performance of the dynamic pricing experiment, while Fig. 6ii depicts the performance of the dynamic pruning experiment. In both experiments, we set the number of rounds in the elimination phase to be a multiple factor of the initial number of clusters (c). In the case of dynamic pricing, we set the number of clusters to 16, and in the case of dynamic pruning, we set the number of clusters to 32. Since κ does not impact the performance of the SBE algorithm, we exclude SBE from this set of experiments.

The analysis reveals that the misidentification probability does not follow a specific trend as a function of the κ value. This observation suggests a nuanced relationship between elimination rounds and algorithm performance. Our hypothesis posits that keeping the number of rounds to the elimination phase too small is not recommended, as this premature elimination can degrade the performance of all elimination-based algorithms because a limited number of samples might not provide sufficient information to make accurate decisions about which actions to eliminate. An elimination decision based on very few samples can result in the elimination of potentially ‘good’ blocks. Conversely, excessively extending the number of rounds in the elimination phase does not notably enhance performance. This is because prolonging the elimination process tends to include too many suboptimal or ‘bad’ actions, leading to inefficiencies in the algorithm.

For algorithms like **ReBEL_{se}** and **ReBEL_{TT}**, we observe that misidentification probability initially decreases as the number of samples in the elimination phase increases. This trend continues up to a certain threshold, beyond which the misidentification probability begins to increase. This pattern aligns with our hypothesis and underscores the existence of an optimal balance in the number of samples required during the elimination phase to maximize performance effectively. In the case of the **ReBEL_{hi}** algorithm, no consistent trend is discernible in terms of κ value adjustments. However, in general it appears that setting the κ value to approximately 1.5 to 2 times the initial cluster size tends to yield the best results.

In our final sensitivity analysis, we examine how the number of samples collected in each round impacts the performance of the proposed algorithms. This study is critical in understanding the reliability of the estimates which directly influences the effectiveness of the algorithms in identifying optimal and eliminating suboptimal actions.

In the dynamic pricing scenario, each round involves presenting prices to 100 customers. The revenue generated from each price pair during that round is averaged to estimate the revenue for that particular price setting. It is reasonable to suggest that increasing the number of customers per round would enhance the reliability of the revenue estimation. A larger sample size tends to reduce the variance in the estimate, thereby providing a more accurate reflection of the true mean revenue. This enhanced accuracy can significantly aid the algorithm in swiftly identifying the most lucrative price points and discarding less profitable options.

In the dynamic pruning case, pruning ratio triplets are tested 25 times on the same network in each round, with the average network gain being calculated subsequently. Similar to Case I, increasing the number of trials per round would presumably lead to a more accurate estimation of the mean network gains. This is because a greater number of trials diminishes the impact of anomalies or outliers on the average gain, leading to a more stable and reliable estimation.

In both cases, a higher number of samples per round allows for a more robust and accurate estimation of mean rewards, which is vital for the algorithms' efficiency. More reliable estimates help the algorithm distinguish more effectively between the performance of different blocks or actions, facilitating faster identification of the optimal blocks and more accurate elimination of the suboptimal ones. This can be observed in the plots in Figs. 8i, ii, where the misidentification probability is plotted against the number of customers (dynamic pricing) and the trials per round (dynamic pruning), respectively.

5 Conclusions

We contribute with a class of Thompson sampling-based recursive block elimination algorithms for dynamic assignment in pure-exploration setting with a limited budget. Our algorithm begins by discretizing the continuous action space into a set of finite discrete actions, followed by the recursive elimination of suboptimal action blocks. Additionally, we implement a zooming mechanism that further clusters these action blocks recursively and eliminate. This elimination process involves calculating confidence bounds over the blocks. We have proposed two distinct techniques for estimating these confidence bounds. Moreover, we have explored variants of the proposed algorithms that employ top-two sampling. We conducted extensive experiments across two problem settings, demonstrating that our approach achieves a lower misidentification probability compared to state-of-the-art algorithms. We also examined the sensitivity of the algorithms to various hyperparameters. Our sensitivity analysis indicates that the proposed algorithms are robust across a range of hyperparameter settings.

For future work, we plan to conduct a regret analysis of the proposed algorithms and explore their extension to adversarial settings.

Acknowledgements This work is partially funded by the EU Horizon Grant 'Integration and Harmonization of Logistics Operations' TRACE (#101104278). The authors would also like to thank the Saudi Prime Minister (Crown Prince Mohammad Bin Salman), the Royal Saudi Air Force, Major General Abdulmonaim ALharbi, Senior Engineer Mohammad Aleissa, Saudi Arabia, and the Saudi Arabian Cultural Bureau in the UK for their support and encouragement.

Author contributions S.P.P wrote the main manuscript. C.A and S.A helped with the experiments and plots. All authors reviewed the manuscript.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- (2023) Closed-Loop Data Science Interaction with Probabilistic Models. <https://www.gla.ac.uk/schools/computing/research/researchsections/ida-section/closedloop/>. Accessed: 2023-11-29
- Agrawal S, Goyal N (2012) Analysis of thompson sampling for the multi-armed bandit problem. In: Conference on Learning Theory, 39–1. JMLR Workshop and Conference Proceedings
- Agrawal S, Goyal N (2013) Thompson sampling for contextual bandits with linear payoffs. In: International Conference on Machine Learning, pp. 127–135. PMLR
- Audibert J-Y, Bubeck S (2010) Best arm identification in multi-armed bandits. In: COLT-23th Conference on Learning Theory-2010, p. 13
- Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. *Mach Learn* 2:235–256
- Auer P, Cesa-Bianchi N, Freund Y, Schapire RE (2002) The nonstochastic multiarmed bandit problem. *SIAM J Comput* 32(1):48–77
- Ban Y, He J (2020) Generic outlier detection in multi-armed bandit. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 913–923
- Breugelmans E, Campo K (2016) Cross-channel effects of price promotions: an empirical analysis of the multi-channel grocery retail sector. *J Retail* 92(3):333–351
- Bubeck S, Munos R, Stoltz G (2011) Pure exploration in finitely-armed and continuous-armed bandits. *Theoret Comput Sci* 412(19):1832–1852
- Bubeck S, Munos R, Stoltz G, Szepesvári C (2011) X-armed bandits. *J Mach Learn Res* 12(5)
- Carpentier A, Locatelli A (2016) Tight (lower) bounds for the fixed budget best arm identification bandit problem. In: Conference on Learning Theory, pp. 590–604. PMLR
- Cesa-Bianchi N, Lugosi G (2006) Prediction, Learning, and Games. Cambridge University Press. <https://doi.org/10.1017/CBO9780511546921>
- Chapelle O, Li L (2011) An empirical evaluation of thompson sampling. *Adv Neural Inf Process Syst* 24
- Chawla S, Gionis A (2013) k-means-: A unified approach to clustering and outlier detection. In: Proceedings of the 2013 SIAM International Conference on Data Mining, pp. 189–197. SIAM
- Coquelin P-A, Munos R (2007) Bandit algorithms for tree search. In: Uncertainty in Artificial Intelligence
- Faella M, Finzi A, Sauro L (2020) Rapidly finding the best arm using variance. In: ECAI 2020, pp. 2585–2591. IOS Press
- Gabillon V, Ghavamzadeh M, Lazaric A (2012) Best arm identification: A unified approach to fixed budget and fixed confidence. *Adv Neural Inf Process Syst* 25
- Gorodnichenko Y, Sheremirov V, Talavera O (2018) Price setting in online markets: Does it click? *J Eur Econ Assoc* 16(6):1764–1811
- Hult GTM, Sharma PN, Morgeson FV III, Zhang Y (2019) Antecedents and consequences of customer satisfaction: do they differ across online and offline purchases? *J Retail* 95(1):10–23
- Jamieson K, Talwalkar A (2016) Non-stochastic best arm identification and hyperparameter optimization. In: Artificial Intelligence and Statistics, pp. 240–248. PMLR
- Jiang Z, Xu Y, Xu H, Wang Z, Qiao C, Zhao Y (2022) Fedmp: Federated learning through adaptive model pruning in heterogeneous edge computing. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE), pp. 767–779. IEEE

- Jourdan M, Degenne R, Baudry D, Heide R, Kaufmann E (2022) Top two algorithms revisited. *Adv Neural Inf Process Syst* 35:26791–26803
- Karnin Z, Koren T, Somekh O (2013) Almost optimal exploration in multi-armed bandits. In: *International Conference on Machine Learning*, pp. 1238–1246. PMLR
- Kleinberg R, Slivkins A, Upfal E (2019) Bandits and experts in metric spaces. *J ACM* 66(4):1–77
- Kleinberg R, Slivkins A, Upfal E (2008) Multi-armed bandits in metric spaces. In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pp. 681–690
- Lattimore T, Szepesvári C (2020) *Bandit Algorithms*. Cambridge University Press. <https://doi.org/10.1017/9781108571401>
- Li L, Chu W, Langford J, Schapire RE (2010) A contextual-bandit approach to personalized news article recommendation. In: *Proceedings of the 19th International Conference on World Wide Web*, pp. 661–670
- Long Q, Anagnostopoulos C, Parambath SP, Bi D (2023) Feddip: Federated learning with extreme dynamic pruning and incremental regularization. In: *2023 IEEE International Conference on Data Mining (ICDM)*, pp. 1187–1192. IEEE
- Luo Y, Sun WW, Liu Y (2022) Contextual dynamic pricing with unknown noise: Explore-then-ucb strategy and improved regrets. *Adv Neural Inf Process Syst* 35:37445–37457
- Luo Y, Sun WW et al (2021) Distribution-free contextual dynamic pricing. *arXiv preprint arXiv:2109.07340*
- Magureanu S, Combes R, Proutiere A (2014) Lipschitz bandits: Regret lower bound and optimal algorithms. In: *Conference on Learning Theory*, pp. 975–999. PMLR
- McMahan HB, Streeter M (2009) Tighter bounds for multi-armed bandits with expert advice. In: *Proceedings of the 22nd Annual Conference on Learning Theory (COLT)*
- McMahan B, Moore E, Ramage D, Hampson S, Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR
- Misra K, Schwartz EM, Abernethy J (2019) Dynamic online pricing with incomplete information using multiarmed bandit experiments. *Mark Sci* 38(2):226–252
- Mueller JW, Syrgkanis V, Taddy M (2019) Low-rank bandit methods for high-dimensional dynamic pricing. *Adv Neural Inf Process Syst* 32
- Mussi M, Genalti G, Trovò F, Nuara A, Gatti N, Restelli M (2022) Pricing the long tail by explainable product aggregation and monotonic bandits. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3623–3633
- Podimata C, Slivkins A (2021) Adaptive discretization for adversarial lipschitz bandits. In: *Conference on Learning Theory*, pp. 3788–3805. PMLR
- Puthiya Parambath SA, Al-Fahad SAM, Anagnostopoulos C, Kolomvatsos K (2024) Sequential block elimination for dynamic pricing
- Russo D (2016) Simple bayesian algorithms for best arm identification. In: *Conference on Learning Theory*, pp. 1417–1418. PMLR
- Sebastian Stein RM-S (2023) *Interactive Model-based Probabilistic Visualisations for Exploring Decisions*. Technical report, University of Glasgow, School Of Computing Science (2023)
- Shahrampour S, Noshad M, Tarokh V (2017) On sequential elimination algorithms for best-arm identification in multi-armed bandits. *IEEE Trans Signal Process* 65(16):4281–4292
- Sutton RS, Barto AG (2018) *Reinforcement Learning: An Introduction*. MIT press
- Tirinzoni A, Degenne R (2022) On elimination strategies for bandit fixed-confidence identification. *Adv Neural Inf Process Syst* 35:18586–18598
- Wang S, Zhu J (2022) Thompson sampling for (combinatorial) pure exploration. In: *International Conference on Machine Learning*, pp. 23470–23483. PMLR
- Xu J, Wang Y-X (2022) Towards agnostic feature-based dynamic pricing: Linear policies vs linear valuation with unknown noise. In: *International Conference on Artificial Intelligence and Statistics*, pp. 9643–9662. PMLR
- Zhuang H, Wang C, Wang Y (2017) Identifying outlier arms in multi-armed bandit. *Adv Neural Inf Process Syst* 30
- Zhu Y, Katariya S, Nowak R (2020) Robust outlier arm identification. In: *International Conference on Machine Learning*, pp. 11566–11575. PMLR