Exploring Bandit Algorithms in Sparse Environments

Does increasing the level of sparsity enhance the advantage of sparsity-adapted Multi-Armed Bandit algorithms?

Author: Rafał Owczarski[1]
Supervisor(s): Julia Olkhovskaya[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Rafał Owczarski
Final project course: CSE3000 Research Project
Thesis committee: Julia Olkhovskaya, Dr. Ranga Rao Venkatesha Prasad

Abstract

In sequential decision-making, Multi-armed Bandit (MAB) models the dilemma of exploration versus exploitation. The problem is commonly situated in an unknown environment where a player iteratively selects one action from a set of predetermined choices. The player's choices can be evaluated by comparing observed rewards after each decision to the highest one obtainable from the set of possible actions. The goal is to minimize the measure of *regret* of not choosing the optimal action every time. Intuitively, one might think of exploring all possible options first and then selecting the one for which the highest rewards were observed. However, it is difficult to model this behaviour when the specifications of the environment setting are not known or when it changes in time. Some classifications of environments exist and can be used to decide which approach can be used to model the MAB problem. The literature provides various algorithms used to model different MAB problems but lacks comparisons between proposed algorithms across the domains they aim to solve. To fill this gap, we want to focus on several Multi-armed Bandit algorithms (policies) and compare their effectiveness and optimality in different environments. This research focuses on a class of environments with a sparse reward function, where the reward depends on the action-specific contextual information and some sparse vector that dictates which features are considered. Four MAB algorithms were selected, each developed to solve the problem in a different environment. The comparison was made in a series of experiments to explore how sparsity affects their performance. The experiments conclude that the advantage of using sparsity-adopted algorithms depends on both sparsity and the environment setting, but increasing the sparsity helps achieve better results than traditional methods.

1 Introduction

Multi-armed bandit problems were first studied by H. Robbins in 1952. The fundamentals were introduced in *Some aspects of the sequential design of experiments*. In such experiments, the size and composition of the samples are not determined beforehand but measured as a function of the observations performed during the experiment [1]. Since then, multiple approaches have been proposed to model the exploration versus exploitation dilemma.

Customarily, MAB problems can be described as a sequential game set in an environment unknown to the player who has to iteratively choose an action (arm) at a given time A_t based on some observed features of a finite number of arms. In each iteration as a result of choosing A_t, some reward X_t is observed. The algorithm intends to maximize the cumulative reward over *n* iterations $= \sum_{\mathbf{t=1}}^{\mathbf{n}} \mathbf{X_t}$.[2]

In exploring Multi-armed Bandit (MAB) problems, understanding the reward generation framework is crucial, as it influences the selection strategies employed by the algorithms. In the case of *Stochastic Bandits*, the reward is drawn from an unknown, fixed, arm-specific distribution. This setting can be used to model a real-life scenario of repetitive gambling on a slot machine. *Adversaral Bandits* describe a dynamic environment where rewards, possibly influenced by external factors such as player's actions, are determined by an arm-specific function of time ($f_a : \mathbb{N} \rightarrow [0, 1]$). This setting can be used to simulate environments where players' actions affect the rewards observed in the future. The extension to the standard MAB is the class of problems known as *Linear Contextual Bandits*, which incorporate additional contextual information about each arm in the form of feature vectors. In this environment, the reward is a function of the inner product of the observed context. This setting is applicable in systems like personalized recommendations where decision-making is enhanced by contextual information observed. [3].

This work focuses on one specific environment extending Contextual MAB, where the reward is described as a fixed sparse reward function with noisy observations and where the context is i.i.d. In this environment, each arm *A* has a context vector associated with it $C_A \in \mathbb{R}^d$ drawn from an unknown distribution. The expected reward is expressed as a function μ of the inner product $C_A^\top \beta^*$ for a fixed and unknown vector $\beta^* \in \mathbb{R}^d$. The unknown β^* is *sparse*, meaning that the number of non-zero coefficients is much less than the dimension of the vector. Moreover, in such an environment, each reward observation has some zero-mean noise with a constant variance associated with it.

The sparse environment describes many real-world problems, such as reinforcement learning or recommender systems. In these environments, contextual information can contain, for instance, the history of previous user interaction with the system, user preferences, or representation of cookies gathered. These feature representations can be multidimensional, but only a small subset of the features is relevant to the reward obtained. Non-sparse environments can evolve into sparse ones over time when more data is gathered and used as contextual information, but it does not affect the reward determination. A solution that was determined to perform well in a non-sparse environment with low-dimensional feature representation can perform worse in an environment with higher sparsity as it cannot assess which features are taken into account when determining the reward as well as the sparsity-adopted solution.

This encourages the exploration of sparsity-adapted MAB algorithms, which are created to leverage the nature of the reward function to increase effectiveness. This work aims to answer the question: *Does increasing the level of sparsity enhance the advantage of sparsity-adapted Multi-Armed Bandit algorithms?* By investigating this question, we can refine our understanding of MAB algorithms in a sparse environment, enabling us to choose the most adequate algorithm in the environment we want to explore. We will focus on four algorithms designed to operate in different experiment settings and compare them with each other in the environment with fixed sparse reward function and i.i.d context. The exact

algorithms are as follows:

1. Stochastic bandits: Upper Confidence Bound Algorithm (UCB): section 7.1, Algorithm 3 in [2],
2. Adversarial bandits: Exponential-weight algorithm for Exploration and Exploitation (EXP3): [4],
3. Linear contextual bandits: Algorithm 1 in [5],
4. Sparse bandits: namely Sparsity-Agnostic Lasso Algorithm [6],

The later sections briefly introduce the algorithms and their theoretical behaviour. This is followed by exploring each algorithm in a series of experiments described in the methodology section. Then, the results of the experiments are discussed, followed by a thorough analysis of the findings. Finally, concluding the paper, we summarize our contributions, discuss implications, and suggest directions for future research.

2 Methodology, Background, Problem Description

This section describes each algorithm in detail and introduces our experiment setting, explaining how the research question will be answered.

2.1 Formal Problem Description

Multi-armed contextual bandits are characterized by the number of arms - *K* - where each arm is denoted as *i* with $1 \leq i \leq K$. The number of iterations is called a *horizon*; denoted as *T*. At any given time t, the player observes the *d* dimensional contextual information on each of the arms $C_{t,i}$, selects an arm A_t and observes some reward $X_{A_t}(t)$.

The comparison of the UCB, EXP3, linUCB, and Sparsity Agnostic Lasso Bandit algorithms highlights their different approaches to multi-armed bandit problems. Each algorithm has unique advantages, such as balancing exploration and exploitation, handling adversarial settings, using contextual information, or managing sparse rewards. To assess their performance fairly, we use uniform metrics that are transferable across environments.

The performance of the algorithms is evaluated using a measure called *regret*. It describes the performance of the algorithm in comparison to the reward observed on the optimal arm at time *t* - A_t^*. Intuitively, maximizing the cumulative reward is equivalent to minimizing the regret value. Formally, this measure at time *T* is defined as:

$$\mathcal{R}^{\pi}(T) := \sum_{t=1}^{T} \mathbb{E}[\mu(C_{t,A_t^*}^{\top}\beta^*) - \mu(C_{t,A_t}^{\top}\beta^*)]$$

[6]

To assess the complexity, we also aim to compare the running times of each algorithm and the memory accessed during each experiment.

Four algorithms have been selected for the experiment. They differ in complexity and the nature of the problem they aim to solve. They will be formally introduced with the exact pseudocode available in the referenced works.

UCB at a time *t* selects arm A_t that maximises the upper confidence bound measure:

$$UCB_i(t-1,\delta) = \begin{cases} \infty & \text{if } T_i(t-1) = 0 \\ \hat{\mu}_i(t-1) + \sqrt{\frac{2\log(1/\delta)}{T_i(t-1)}} & \text{otherwise} \end{cases}$$

dependent on $T_i(t-1)$ historical observations of reward with empirical mean $\hat{\mu}(t-1)$ on a given arm and an input parameter δ describing the error probability. After the selection, it observes the reward associated with A_t and updates the upper confidence bound. The algorithm aims to ensure optimism that the arm with the highest empirical mean is indeed the optimal one and, therefore, not to explore suboptimal arms unreasonably often. [2]

EXP3 selects arm A_t randomly, based on weighted probabilities obtained by leveraging the estimator. The intuition is that by exponentially increasing the weight of the arm based on the reward received, relative to the likelihood of selecting it, the algorithm ensures that arms that are infrequently selected can rapidly become more likely to be chosen if they return high rewards. In contrast to UCB, the algorithm can be used to model Adversarial Bandit problems that do not assume that rewards come from a stationary distribution.[4]

linUCB is adapted to solve Linear Contextual Bandits. Unlike UCB, it considers additional contextual information available at each round for each arm in the decision-making process. The assumption is that the expected reward for choosing an arm is a linear function of its observed features combined with some unknown coefficients. The algorithm models the problem using the confidence bound around the estimated reward calculated using the least squares estimate of the unknown coefficients. In [5], it was proven that the regret obtained by the algorithm is bounded by $\mathcal{O}(\sqrt{Td\ln^3(KT\ln(T)/\delta)}$.

Sparsity Agnostic Lasso bandit is a sparsity-adapted MAB algorithm that, unlike other similar algorithms, does not require the knowledge of the sparsity of the reward function. The algorithm utilizes *L1* regularization, commonly known as Lasso (Least Absolute Shrinkage and Selection Operator), to introduce an estimator of β^* - denoted as $\hat{\beta}$ - that aims to estimate the unknown coefficient and selects the arm that maximizes the inner product of the context and the β^*.

Lasso regression applies a regularization process where it puts a constraint on the sum of all values of the features taken into account, penalizing the coefficients of the regression variables and effectively shrinking some of them to 0. By increasing the λ parameter of L1 regularization over time, we ensure that multiple coefficients are shrunk to exactly zero, obtaining a sparse vector that can be used to reduce the dimensionality and to estimate β^* [7].

Oh et al. in [6] showed that the regret bound of the algorithm for the two-armed case is $\mathcal{O}(s_0\sqrt{Tlog(dT)}$ and it outperforms other sparsity-adapted algorithms that require prior knowledge of s_0 in numerical experiments.

2.2 Methodology

The study adopts a structured approach to evaluating and comparing the performance of various Multi-armed Bandit

algorithms. This section outlines the general methodology employed to ensure a comprehensive analysis.

Each algorithm was assessed based on its ability to efficiently and accurately identify optimal actions by measuring its regret within environments characterized by varying degrees of sparsity. The methodology can be divided into data generation and algorithm evaluation.

Firstly, synthetic datasets were generated. The datasets mathematically modelled environments with sparse reward derived from the contextual information. The dataset also contained some noise associated with each observation. This ensured that the algorithms were tested in environments resembling real-world situations where data can be sparse and noisy.

Each experiment was run using the synthetic dataset in a separate environment characterized by:

- Horizon T: The number of rounds over which the bandit game was played.
- Number of Arms K: The total number of arms available for selection.
- Context Dimensions d: The dimensionality of the contextual information available at each round.
- Sparsity s_0: The amount of non-zero coefficients in the β^* vector used to generate the reward,
- Context Distribution: The probabilistic distribution that modelled the context generation.

In the evaluation phase, the data obtained from the experiment was compiled and analyzed to find patterns and insights in algorithm performance. Key indicators included the convergence speed to optimal actions, the total cumulative regret, and the computational efficiency (memory usage and running time).

3 Experimental Setup

This section describes the experimental setup used to evaluate the performance of the multi-armed bandit algorithms. We outline the parameters, conditions, and metrics that fairly compare various simulated environments to answer the imposed research question.

The environments used in the algorithms consisted of *K* sparse arms, each associated with i.i.d context. Each run of an experiment was repeated ten times, and the results were averaged. The initially selected experiment variables were chosen based on work mentioned in [6], but additional experiments were run with data not provided in the above-mentioned research to analyze the behaviour of the algorithms.

To observe the relation between varying sparsity and the performance of the algorithms, the sparsity measure s_0 was fixed, and other variables were changed to observe the behaviour changes. In the experiments we used $s_0 = 5$, $K \in \{20, 100\}$ and $d \in \{5, 10, 20, 50, 100\}$. To align with theoretical analysis in [6] that was performed on symmetrically distributed contexts, the contexts were drawn using a multivariate normal distribution with the covariance matrix being $0.5I_d$ and the mean of each coefficient set to 0. To extend the scope of the research, the algorithms were also compared in environments with context drawn from non-symmetrical distributions, namely $X \sim Exp(3)$ and *skew-normal* distribution with skewness equal to 3 and the same mean and covariance matrix as with normally distributed context. The β^* parameter was drawn where each non-zero entry followed the uniform distribution $X \sim \mathcal{U}(0, 1)$ and was then normalized to ensure that the inner product of β^* and C_t falls in range $[0, 1]$. To generate the reward the identity function was used where $Y = \langle X, \beta^* \rangle$ with $0 \leq Y \leq 1$. The algorithms were supplied with observed $Y + \epsilon$ where $\epsilon \sim \mathbf{N}(0, 0.0001)$ and represents the noise (some experiments were conducted with $\epsilon \sim \mathbf{N}(0, 0.01)$ but it was deemed that the noise only influences the variance of the algorithms, not the average cumulative regret bound).

After initial comparison, some more additional analysis was performed between SALasso and LinUCB in an environment with fixed context dimension $d = 50$, number of arms $K = 20$ and varying $s_0 \in \{5, 10, \ldots, 45, 50\}$ with the context being drawn using a multivariate normal distribution with mean 0 and 1 and covariance matrix equal to $0.5I_d$

Before performing any evaluation, the hyperparameters of all algorithms were fine-tuned to the best-performing ones in the current environment by running the algorithms with various values of the hyperparameters against each other in the experimental environment and selecting the hyperparameter that yielded the minimal regret averaged over ten repetitions. The comparison described in the results section was made using the selection of algorithms with the optimal hyperparameters for each algorithm under a given environment.

The hyperparameters used to generate all the figures except for Figure 6 are as follows: Exp3 $\gamma \in \{0.1, 0.4, 0.6, 0.9\}$, LinUCB $\alpha \in \{0.001, 0.01, 0.1, 0.2, 0.4, 0.8\}$, SALasso $\lambda_0 \in \{0.0001, 0.01, 0.1, 0.3\}$. To generate Figure 6, the range of used hyperparameters was extended to ensure more precise numerical results and is as follows: LinUCB $\alpha \in \{0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.8, 0.9\}$, SALasso $\lambda_0 \in \{0.00001, 0.0001, 0.001, 0.01, 0.1\}$.

`SMPyBandits` was used as a framework to ensure consistency between other works in the field and to provide us with benefits such as parallel execution of the experiments [8]. However, the current framework version did not support contextual bandits, so the code must be adjusted for our purposes, the codebase used to generate the graphs can be found in [9]. All environments were modelled using functionalities provided by `numpy` [10]

4 Results and Discussion

The experiment's results can be seen in Figure 1 and Figure 2. The figures contain graphs comparing averaged cumulative regrets obtained in ten repetitions of the experiment with s_0 and varying context dimensions. To represent variability across runs, the graph contains errors with a magnitude equal to 2σ to ensure statistical significance.

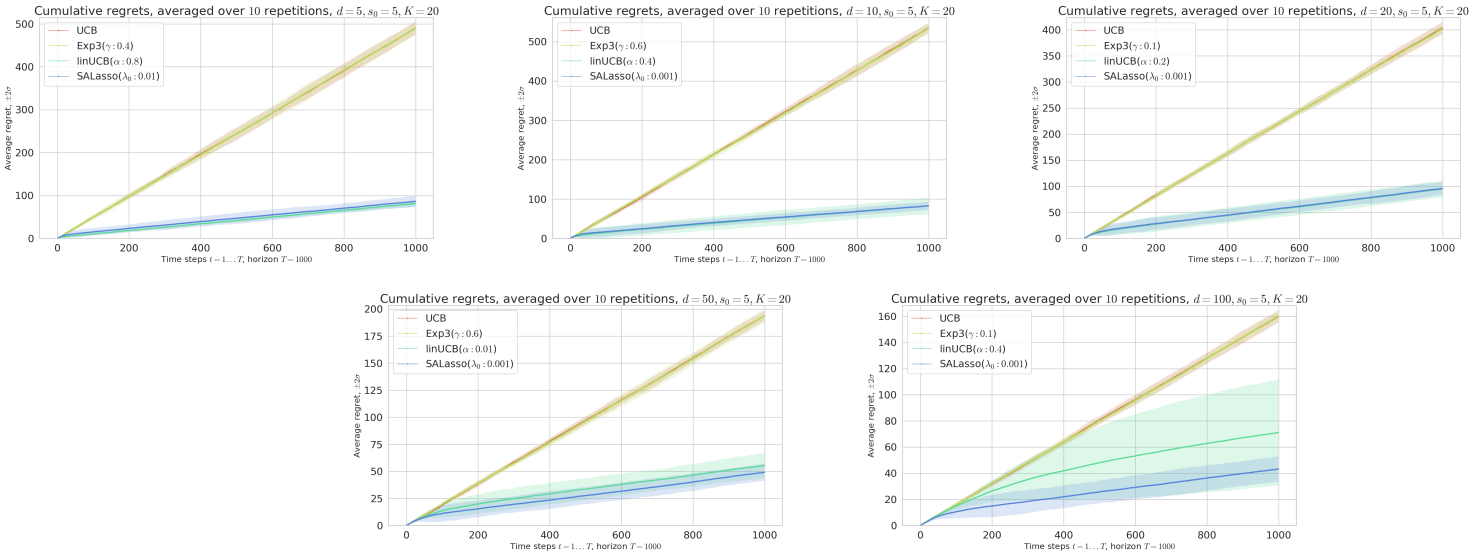

Figure 1: The comparisons of average cumulative regret between UCB, EXP3, LinUCB and Sparsity Agnostic Lasso algorithms in the environment with 20 arms, and $s_0 = 5$ and context with various dimensions

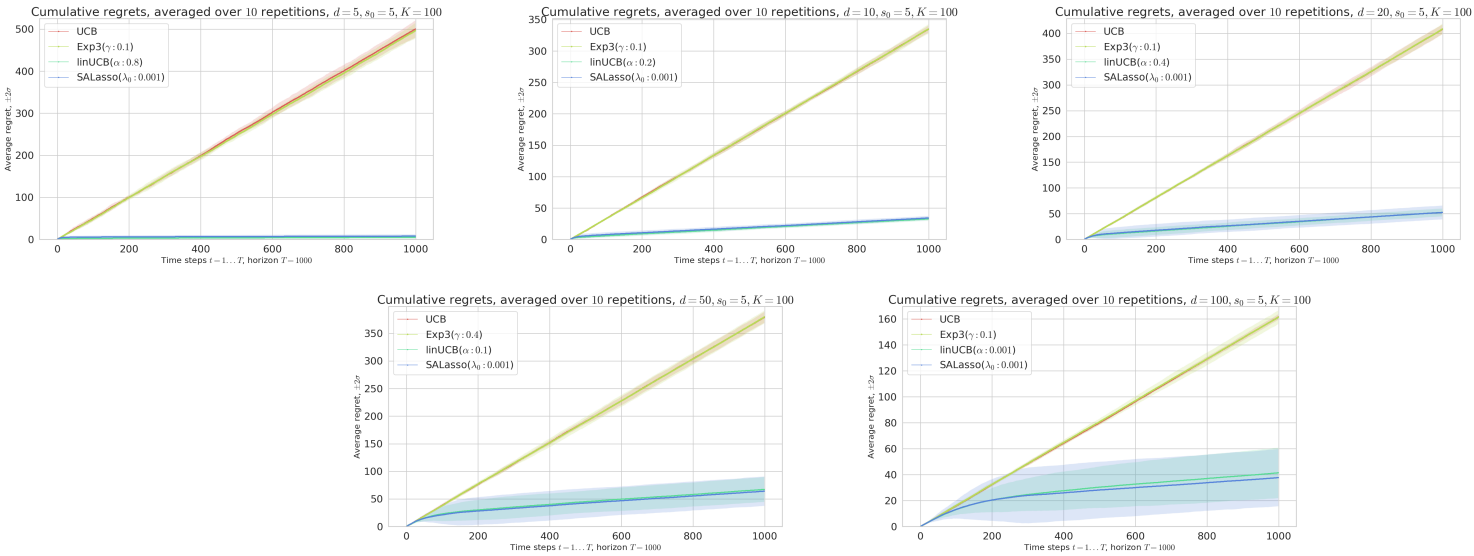

Figure 2: The comparisons of average cumulative regret between UCB, EXP3, LinUCB and Sparsity Agnostic Lasso algorithms in the environment with 100 arms, and $s_0 = 5$ and context with various dimention

The results obtained with symmetrically distributed context show that increasing the proportion of sparsity measures to the context dimension affects the performance of all algorithms. The most notable observation is that the standard deviation of contextual algorithms increases with increasing context dimensions. Both algorithms leveraging contextual information perform similarly in this environment setting, obtaining mean cumulative regrets within the 2 standard deviations of each other. However, SALasso reliably can achieve lower regret bounds in more sparse settings (with $d > 10$). It is important to note that the variability of both algorithms is similar in the environment with $K = 100$, but SALasso achieves smaller variability with $K = 20$. This aligns with the expected upper bounds of regret of LinUCB and SALasso bandit given that the regret obtained by LinUCB is proportional to the number of arms and the dimension while regret bound for SALasso is independent to K.

Conversely, results obtained in environments with non-symmetrical context (showcased in Appendix B and Appendix C) suggest that with $d \geq 50$ SALasso statistically outperforms LinUCB, for $5 < d < 50$ they perform statistically identically and with $d = 5$ LinUCB achives the lowest regret. Notably, the variance in these environments is smaller than in an environment with a symmetrically distributed context. The explanation is that the zero-mean multivariate normal distribution allows negative context coefficients to be drawn as opposite to the asymmetrical distributions with the parameters used in other experiments. This leads to variability in estimating the β^* as similar results can be obtained by estimating negative coefficients on negative context values and positive coefficients on positive context values. Both LinUCB and SALasso showcase this issue. By shifting the mean of Normal distribution used to generate the context to make it positive one can achieve results similar to ones obtained in environments with asymmetrical contexts, where cumulative regrets of LinUCB and SALasso do not lay within 2σ of each other. However, increasing the noise variance to 0.01 enables this situation to persist in these environments too.

It is noticeable that both contextual algorithms converge to logarithmic regret bounds while UCB and EXP3 failed to do so under the horizon constraints and maintained regret linearly proportionate to time. This is explainable because non-context-adapted algorithms aim to determine the optimal arm by searching for the one that returns the highest observable mean - the decision to pull an arm is made purely on past performance metrics. In the used environment, the algorithms struggle to find the optimal arm because of the lack of insight into contextual information, which creates an environment similar to an adversarial environment in which the optimal arm can change over time.

The intuition would then suggest that EXP3 should outperform UCB as it is better adapted to the adversarial reward model. However, this is not the case as both algorithms act seemingly randomly. The K parameter, especially, influences this randomness by minimizing the exponential weight increase benefit of the EXP3 algorithm.

The results showcased in Figure 1 and Figure 2 represent the data obtained with optimal hyperparameters, fine-tuned for specific environments. It is important to note how sensitive all algorithms are to their input hyperparameters. Such variability can greatly influence the real-life usability and efficiency of the algorithms, as in the online setting, it is hard to tune the parameters. In Figure 3, we visualise an example of an experiment run with different input parameters of EXP3, LinUCB and SA Lasso algorithms

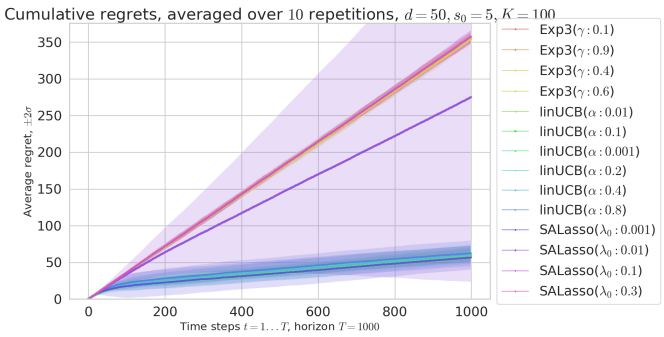

Figure 3: Experiment results for various hyperparameter values

As shown the performance of the SALasso algorithm in this particular environment is highly dependent on the initial parameter λ_0 compared to the other algorithms and their respective hyperparameters. While SA Lasso with $\lambda_0 = 0.001$ performs the best, changing the value to $\lambda_0 > 0.01$ changes the outcome to the worst performing. This is caused by the penalty imposed by the L1 regularization. The λ parameter used at each iteration in Lasso regression depends on time and the initial λ_0. If the hyperparameter is too high, the algorithm under-fits by shrinking too many coefficients to 0 and possibly discarding important features. Smaller values of λ_0 enable the initial exploration phase to gather enough data that can be used over time, when the regularization parameter increases, to more accurately estimate the $beta^*$.

This is not visible in the case of LinUCB, where all versions of the algorithm managed to converge with $T = 1000$. However, the standard deviation of the measurements is influenced by varying the α parameter of LinUCB. This can be explained by the effect of the hyperparameter on the behaviour of LinUCB - the smaller value of α results in more exploitation, whereas a larger one in more exploration. By increasing the exploration we promote more uncertainty, the algorithm tries a larger range of actions looking for the optimal one but is not certain that such action will be found. Diverse choices lead to higher variance in the regret obtained. In one repetition, the algorithm may be able to find an action that minimizes the regret in the long run, but in another, it might fail to do so. This reasoning is not limited to LinUCB, in the case of all bandit algorithms, low cumulative regret in the long run may be associated with a high variance if the exploration is high. On the other hand, too high exploitation can lead to higher cumulative regret bound but lower variability.

Another metric we considered is the algorithms' running time. Figure 4 shows the results of executing $T = 1000$ iterations by all the algorithms. The actual running times can be machine-dependent, but the algorithms' ranking should be

reproducible.

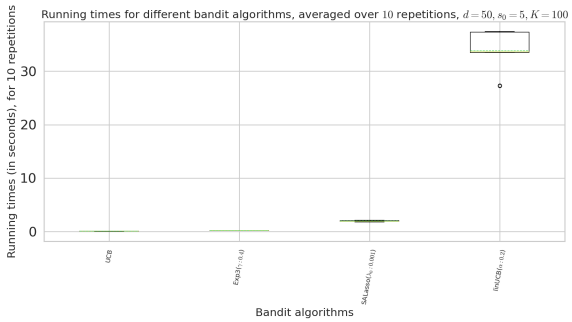

Figure 4: Running times of bandit algorithms

The graph demonstrates the inherent complexity of the algorithms. LinUCB, which involves more complex computations, including matrix inversions and handling higher-dimensional data, naturally shows longer running times. Specifically, the update formula for LinUCB includes inverting the matrix A. Matrix inversion is computationally expensive, especially as the dimensionality d of the context vectors and the number of arms K increase. In contrast, simpler algorithms like UCB and EXP3, which involve straightforward calculations, mostly on scalars, showcase much faster execution times.

UCB and EXP3 not only perform faster but also show little variation across runs, indicating that they are less sensitive to the nuances of different experimental runs or data variations, this is caused by the fact that they usually fail to select the optimal arm, therefore obtaining regret close to the maximum. LinUCB's variability suggests it may be more sensitive to specific data characteristics or experimental setups. On the other hand, SA Lasso bandit combines both of the advantages, showcasing both smaller variance and short running time in comparison to LinUCB

Finally, we tracked the algorithms' memory consumption. Figure 5 showcases the boxplot of averaged memory used in one experiment run in KiB.

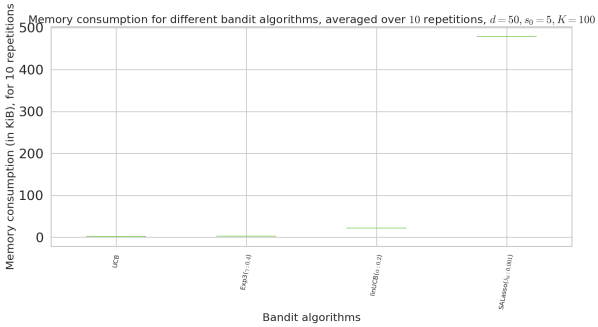

Figure 5: Memory consumption of bandit algorithms

The memory usage was calculated based on approximating the memory used by the algorithm by serializing the objects to a string using `pickle.dumps()` method, which can lead to overestimating the memory used for objects with references to other objects, but in our case, the serialization was performed on policies that only store matrices, vectors or scalars. The memory usage represents the memory used to store the state information for each of the algorithms, not the one accessed when evaluating the choices made at the time t. The memory consumption for all algorithms is relatively low, with none exceeding 1 MB. The obvious outlier is SA Lasso Bandit, which also creates `sklearn.linear_model.Lasso` model to compute the choices.

Since LinUCB and SALasso perform similarly, an additional comparison was made between those two algorithms. Graphs in Figure 6 and Figure 7 obtained cumulative regret of LinUCB compared to SALasso in the environments mentioned in 3.

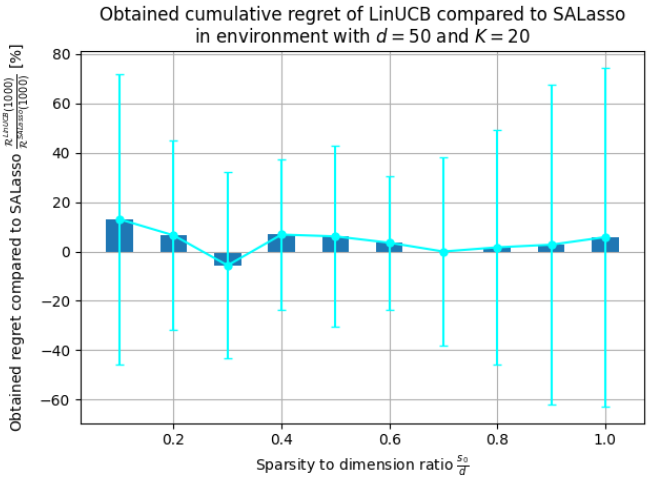

Figure 6: Correlation between ratio of cumulative reward and sparsity to dimension ratio in an environment with multivariate normal context with mean 0

The graph showcases the ratio of cumulative regret obtained by LinUCB to SALasso at $T = 1000$. It is noticeable that, on average, both algorithms perform similarly, with the ratio having high variability. However, while statistically LinUCB can achieve 60% smaller regret than SALasso where $\frac{s_0}{d} \geq 0.9$, SALasso can outperform LinUCB by 80%. As previously discussed, this visualizes how similar both of the algorithms perform in terms of regret and variability in this environment.

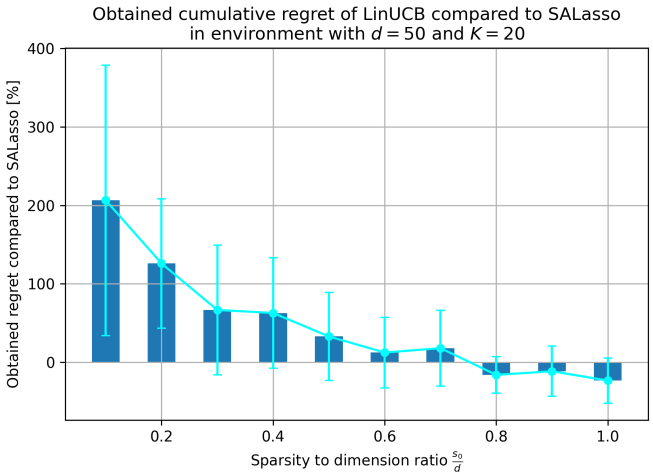

Figure 7: Correlation between ratio of cumulative reward and sparsity to dimension ratio in environment with multivariate normal context with mean 1

The graph depicts the same scenario as Figure 6 but in an environment with less variability, with a multivariate normal context and a mean of 1. It is clearly visible that the lower the $\frac{s_0}{d}$, the higher the advantage of using SALasso, with the algorithm obtaining 200% less regret on average than LinUCB when the ratio is sparsity to context dimension ratio is equal to 0.1. As the sparsity ratio increases, the regret ratio decreases steadily, indicating that LinUCB's performance relative to SALasso improves with lower sparsity, finally outperforming the latter with $\frac{s_0}{d} \geq 0.8$

5 Responsible Research

This section evaluates the ethical considerations and the reproducibility of our experimental methods employed in the analysis of bandit algorithms. We ensured each algorithm was assessed under consistent and fair conditions, using synthetically generated data to avoid privacy concerns while carefully documenting all experimental procedures. This allows us to discuss not only the effectiveness of each algorithm but also the broader implications of their application and the integrity of the research practices.

The algorithms were fairly compared in the exact same environment in each run of the experiment. The context and rewards were pre-computed and cached for each repetition using the same initialization vectors and probabilistic distributions. The vectors were normalized, and the results were averaged over multiple repetitions to evaluate the variability of the results and mitigate the randomness and bias, enhancing the reliability and fairness of our comparisons.

In the experiments, we used synthetic data, which avoided the privacy issues of real-world datasets. However, artificial data does not reflect the complexity of the algorithms' usual applications. Such experiment settings can oversimplify and overlook crucial aspects of datasets obtained from user-interactive systems.

To ensure reproducibility, the hyperparameters, the pseudocode, and the repository used are fully disclosed and referenced. Additionally, the methodology section mentions a detailed explanation of all other variable parameters in the experiment setting, such as d, K, and s_0. With this information, other researchers should be able to replicate our experiments and verify our findings.

By fixing the s_0 measure and only varying the dimension of the context, we were able to test the algorithms' behaviour in environments with no or very high sparsity. This helps us understand each algorithm's limitations under different assumptions and evaluate them in extreme conditions as well as more lenient ones.

The consideration of multiple metrics to evaluate the influence of increasing sparsity helped us understand various aspects of the algorithms along with their performance.

Lastly, by selecting cumulative regret as the primary measure for our analysis, we ensure that our results align with and are comparable to other studies within the field. Cumulative regret is a standardized metric widely used to evaluate the performance of bandit algorithms, which facilitates the benchmarking ability of the results obtained in this research.

6 Future Work

This section discusses the future potential and research opportunities linked to the research question. This section outlines the key areas where our research can be improved and expanded to yield new or more robust results.

While the study highlights the influence of sparsity in controlled artificial environments with simulated noisy observations, the performance of the algorithms in real-life scenarios was not explored in the scope of this research. We could not find a dataset that enables us to compare on-policy algorithms under the time constraints we faced. The main difficulty was the lack of representation of the reward for the actions not chosen by the algorithm that generated the dataset (e.g. ad campaign results). Extending the experiment setup with data obtained from real-life datasets can be beneficial to understand further the benefits of using sparsity-adapted algorithms in real-life scenarios.

In the study, we focused on the SALasso bandit algorithm as a representative of the class of sparsity-adapted bandit algorithms. It was selected because it was shown that it obtains the lowest regret bound out of all algorithms in this class [6]. However, the research would also benefit from other sparsity-adapted bandit algorithms to explore whether any algorithm of that class achieves more optimal regret than the traditional approaches. This can further explain the benefits of using sparsity-adapted algorithms or prove that not all sparsity-adopted algorithms perform better than their traditional counterparts in certain environments. Similarly, the research would benefit from testing other traditional algorithms against the SALasso bandit. This can improve the understanding of which traditional algorithms behave best in contextual sparse environments and can be used when, for some reason, sparsity-adapted algorithms are impossible to deploy.

Another variant we would like to propose is to explore the influence of dimensionality reduction approaches, mainly in the case of LinUCB or other contextual bandit algorithms that are non-sparsity-adopted for their performance in the sparse

setting. The main drawback of using LinUCB is the amount of matrix operations, such as inversion, needed to choose an action at a given timestamp. The complexity of these operations increases proportionally to the dimension of the context. By utilizing techniques such as PCA (Principal Component Analysis) or Random Forrest to reduce the dimension of the context vectors without losing significant information, we can improve the performance and create new algorithms that are more suitable for handling sparse environments.

This research limited the experimental $s_0 = 5$. This enabled test environments with high-dimensional contexts and large and low-dimensional contexts with low sparsity. We noticed that the algorithm's performance changed by altering the number of arms. We believe similar behaviour can be observed when using high-dimensional context and large s_0 value. By fixing $s_0 = 50$ and testing the algorithms using dimensions $d \in \{50, 100, 200, 500, 1000\}$, we should be able to observe results that can give us more insights into the research question. Similarly, one can fix the context dimension and gradually increase s_0 to measure the influence of sparsity with the constant context dimension.

The research focused on comparing the algorithms, assuming that the context is i.i.d distributed and drawn using the Gaussian distribution. However, multiple other approaches exist to draw the context of each arm. As a possible extension to this research, we want to propose an experiment in which the context is drawn from a multivariate Gaussian distribution, with varying correlations between arms and uniform and non-Gaussian elliptical distributions, similar to the experiment mentioned in [6]. This experiment setup was used to determine the superiority of SALasso bandit over other sparsity-adopted algorithms so it can also be leveraged for further comparison of traditional algorithms in the sparse environment. Furthermore, we can also extend the experiment setup with other reward functions except for the identity function.

7 Conclusions

In this research, we aimed to contribute by addressing the research gap in which multi-armed bandit algorithms are not compared to each other in environments that they were not specifically designed to perform. We wanted to quantitatively demonstrate the effectiveness of sparsity-adapted algorithms across different levels of sparsity in comparison to other traditional solutions that do not take sparsity into account.

The primary research question addressed in this study is: *"Does increasing the level of sparsity enhance the advantage of sparsity-adapted Multi-Armed Bandit algorithms?"* This question explores whether algorithms specifically designed to handle sparse data environments perform better as the sparsity of the data increases. Sparsity here refers to situations where only a small number of features or factors are relevant in decision-making, which is common in many high-dimensional data scenarios.

After performing multiple experiments in an environment with a fixed reward function, i.i.d context and increasing sparsity measure in comparison to the dimension of the context vector $d - s_0$ we can conclude that increasing this measure indeed enhances the performance of sparsity-adapted multi-armed bandit algorithms, namely Sparsity Agnostic Lasso bandit algorithm, in comparison to traditional multi-armed bandit algorithms that do not account for sparsity. Statistically, LinUCB can obtain regrets similar to SALasso's under certain conditions, mostly in environments with high variance.

In environments with low variance and with high sparsity ($s_0 = 5, d \in \{50, 100\}$ and 20 arms SALasso demonstrated a lower cumulative regret bound compared to LinUCB, which performs better in lower-sparsity settings ($s_0 = 5, d \in \{5, 10\}$).

It was shown that all algorithms were sensitive to their input parameters, with the SALasso bandit showcasing the highest sensitivity to its hyperparameter λ_0 that can greatly affect the bound to which cumulative regret converges.

The computational complexity analysis revealed that LinUCB operates the slowest in both sparse and non-sparse environments, whereas SALasso makes decisions as fast as less complex algorithms. Conversely, the sparsity-adapted algorithm requires the most memory to facilitate its operation.

Our study shows that sparsity-adapted algorithms like SALasso outperform traditional algorithms in high-sparsity environments. While LinUCB performs well in low-sparsity settings, SALasso consistently shows lower regret in environments with sparse rewards. We showcased that the choice of hyperparameters, especially for SALasso, is critical for optimizing performance, and there exist environments in which both sparsity-adapted and non-sparsity-adopted algorithms perform statistically identically.

8 Acknowledgments

I would like to thank the Supervisor of this Research Project, Julia Olkhovskaya, whose guidance and help throughout the entire process were invaluable. I also extend my gratitude to my peers from the Research Project group who helped develop the code and provided help in countless areas of executing this research. I would also like to recognize the peers who spent their time giving constructive feedback on the paper, making it possible to convey my thoughts and results in an accessible manner.

References

[1] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

[2] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[3] N. Abe and P. Long, "Associative reinforcement learning using linear probabilistic concepts," Jun. 1999.

[4] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM journal on computing*, vol. 32, no. 1, pp. 48–77, 2002.

[5] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 208–214.

[6] M.-h. Oh, G. Iyengar, and A. Zeevi, "Sparsity-agnostic lasso bandit," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8271–8280.

[7] V. Fonti and E. Belitser, "Feature selection using lasso," *VU Amsterdam research paper in business analytics*, vol. 30, pp. 1–25, 2017.

[8] L. Besson, *SMPyBandits: an Open-Source Research Framework for Single and Multi-Players Multi-Arms Bandits (MAB) Algorithms in Python*, Online at: GitHub . com / SMPyBandits / SMPyBandits, Code at https://github.com/SMPyBandits/SMPyBandits/, documentation at https://smpybandits.github.io/, 2018. [Online]. Available: https://github.com/SMPyBandits/SMPyBandits/.

[9] D. Arsene, C. M. Boon, M. Herrebout, W. Hu, and R. Owczarski, *Contextual SMPyBandits*, Online at: GitHub . com / thatCbean / SMPyBandits, 2024. [Online]. Available: https : / / github . com / thatCbean / SMPyBandits/.

[10] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10 . 1038 / s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

A Use of AI in the work

The AI was mainly used to aid the writing of the report.ChatGPT was used to make some complicated sentences more readable and understandable, and Grammarly was used to fix the punctuation and spelling. The prompts used included ”Make this sentence clear and split it into smaller sentences without changing the wording or meaning”. However, the results were unsatisfactory and often edited to match the context of the rest of the text. No AI was used to generate ideas and retrieve information in this research. In the initial part of the research, when the topic was not clear enough, ChatGPT was used to translate the notation of one paper to another (as often research papers used different notations for reward, context, beta/theta, and it was difficult to understand the algorithms and pseudocodes fully with having the notation mixed). The prompt used consisted of a notation definition and part of the paper that needed to be translated.

B Results Obtained in Sparse Environment with Asymmetric Context Distribution

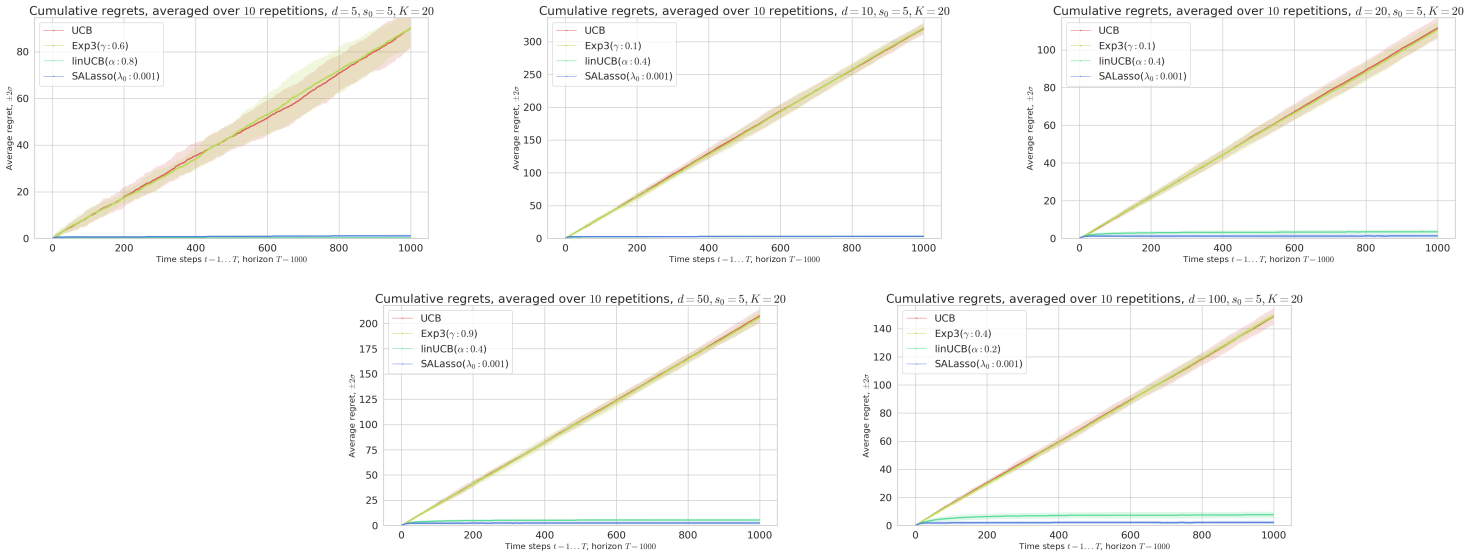

Figure 8: The comparisons of average cumulative regret between UCB, EXP3, LinUCB and Sparsity Agnostic Lasso algorithms in the environment with 20 arms, and $s_0 = 5$ and context distributed using Skewed Normal distribution with various dimensions

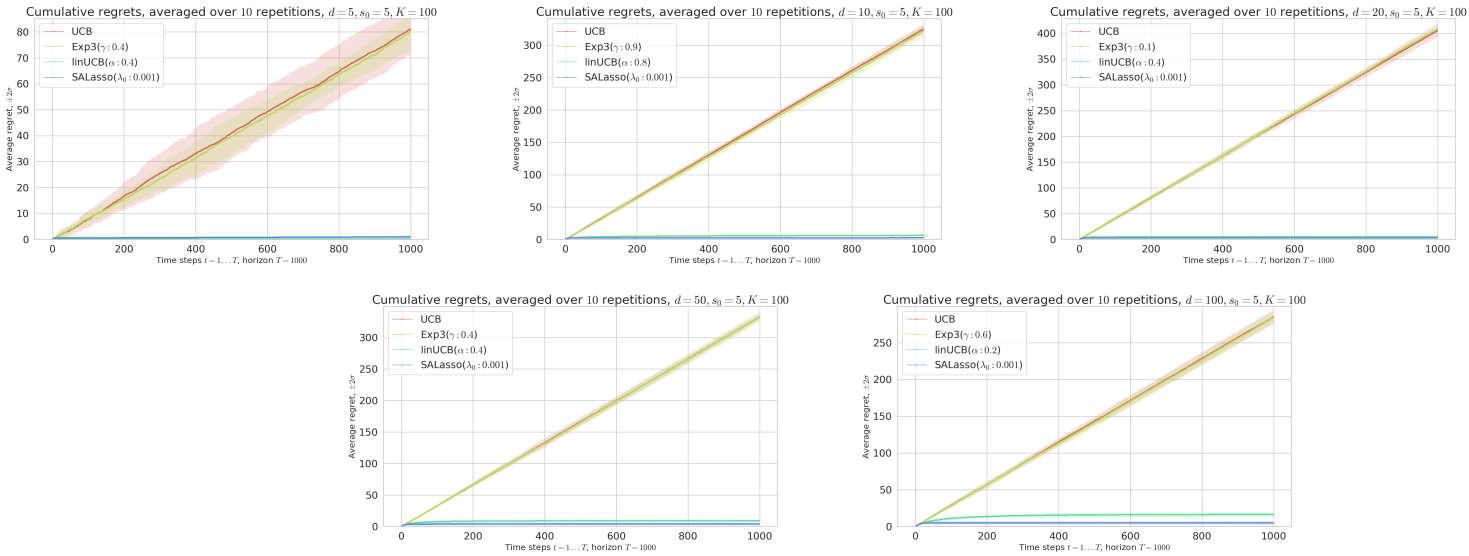

Figure 9: The comparisons of average cumulative regret between UCB, EXP3, LinUCB and Sparsity Agnostic Lasso algorithms in the environment with 100 arms, and $s_0 = 5$ and context distributed using Skewed Normal distribution with various dimentions

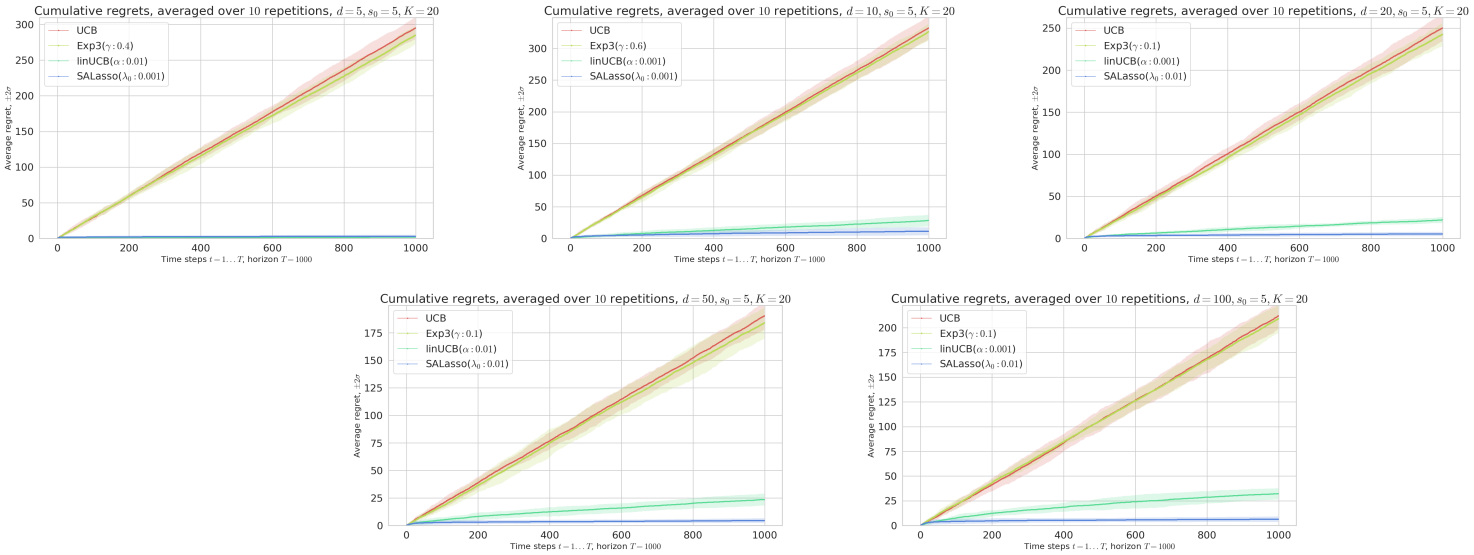

Figure 10: The comparisons of average cumulative regret between UCB, EXP3, LinUCB and Sparsity Agnostic Lasso algorithms in the environment with 20 arms, and $s_0 = 5$ and context distributed using Exponential distribution with various dimensions

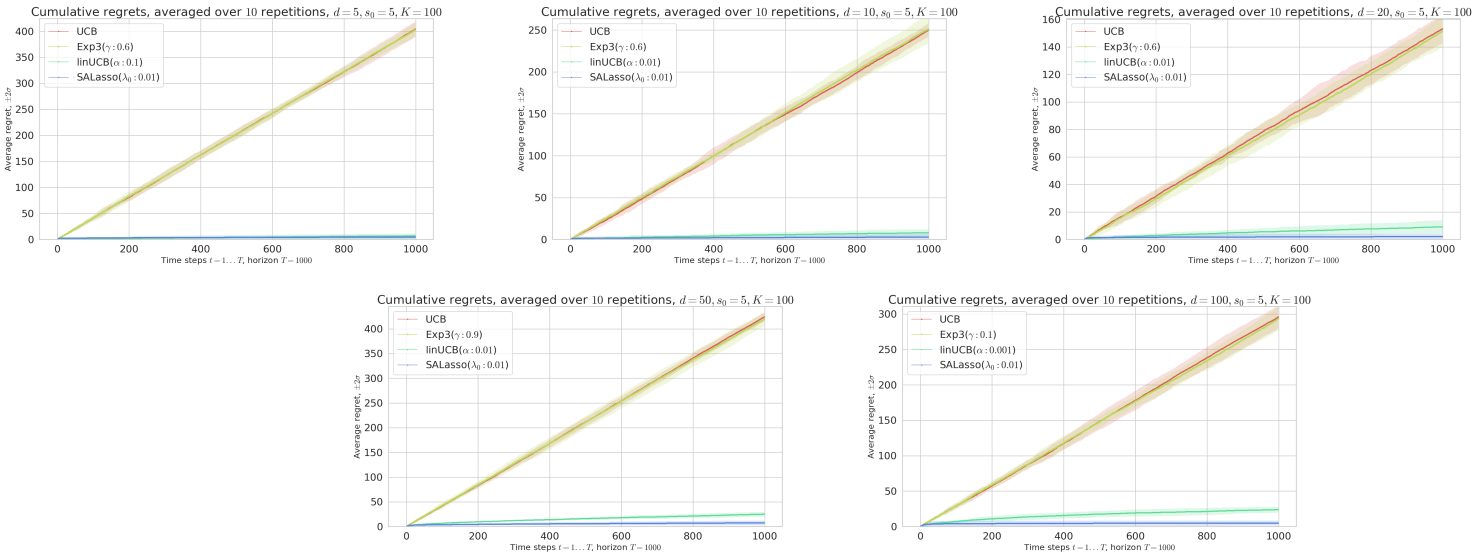

Figure 11: The comparisons of average cumulative regret between UCB, EXP3, LinUCB and Sparsity Agnostic Lasso algorithms in the environment with 100 arms, and $s_0 = 5$ and context distributed using Exponential distribution with various dimentions