Thompson Sampling for Multi-armed Bandit Problems: From Theory to Applications

Yichen Wang*

Reading Academy, Nanjing University of Information Science and Technology, Nanjing 210044, China

*Corresponding author: dy804646@student.reading.ac.uk

Keywords: reinforcement learning, multi-armed bandit, Thompson sampling, decision making.

Abstract: A multi-armed bandit serves as a simple but powerful framework for reinforcement learning algorithms that can make decisions over time under uncertainty. The multi-armed bandit problem involves an interesting topic: the goal is to maximize the payoff through a balance of exploration and exploitation, for which a great deal of work has accumulated over the years. This paper introduces and models the multi-armed bandit problem, and mainly demonstrates three most common multi-armed bandit algorithms, ε-greedy, upper confidence bound, and Thompson sampling (TS). Among them, TS is characterized by solving a wide range of problems in a computationally efficient way and is currently widely used. This paper will introduce three practical applications in different fields. Finally, this paper summarizes the main applicable types of practical problems and the shortcomings of each algorithm and provides some promising directions for future research in related fields. It is concluded that the advantages of TS allow quick and efficient converge to optima in decision-making problems with no prior information, while the disadvantages of it remain open to improve in several aspects.

1. Introduction

Reinforcement learning is a technique of machine learning that is currently considered to be a way to iterate optimal configuration parameters through data fitting so that artificial intelligence can achieve a high recognition rate in a shorter period. Reward maximization [1] and trial-and-error experience [2] are sufficient to develop competent behaviors that exhibit intelligence. Therefore, reinforcement learning, as a branch of artificial intelligence based on reward maximization, can promote the development of general artificial intelligence [3]. Reinforcement learning is a mapping learning of the intelligent system from the environment (everything that interacts with the agent) [4] to the behavior (the choices of actions). By using reinforcement learning, a reinforcing signal is returned as feedback, and action plans are improved to suit the environment. A deep disparity between reinforcement learning and deep learning is the amount of information given for training and the behavioural purpose (evaluation and instruction) for using that information [5]. Taking Google's AlphaGo and AlphaZero as two examples, the former generates a model after training about 30 million sets of human chess data through a deep convolutional neural network in deep learning. The latter uses the reinforcement learning method, which is simply to generate a model by playing chess with yourself. The result of this experiment was AlphaGo defeated the top human players in Go, and Alpha Zero defeated AlphaGo [6].

The purpose of reinforcement learning is to maximize cumulative total of rewards received after each action. And the aim with limited time is to try to choose actions that can maximize the cumulative reward without knowing the environment, that is, the reward probability of each action. On the one hand, it is necessary to ensure the existing reward efficiency, and on the other hand, some exploration is required to see if there is an unknown action with a higher reward. Therefore, a standard reinforcement learning algorithm must include a balance between exploration and exploitation, which means the exploration-exploitation dilemma [7]. Exploration refers to doing things that have never been done before to obtain a higher reward, which helps the agent fully understand its state space.

Exploitation means doing what the agent currently knows will yield the greatest reward, which helps it find the optimal sequence of actions. A goal of reinforcement learning is to maximize the cumulative return, and a variety of algorithms have been applied over the years to this decision process.

A final reward of a reinforcement learning task is observed after an action with several steps. The simplest case is to maximize a one-step reward. A theoretical model corresponding to the single-step reinforcement learning task is named "K-armed bandit" [8] this paper will focus on the multi-armed bandit problem. The multi-armed bandit problem can be traced back to the following scenario in gambling: a gambler in a casino facing a row of identical-looking gambling machines wants to maximize returns without knowing reward probabilities [9]. How to make choices in sequence? MAB problem is also included in many different real-world problems, such as advertisement placement. Figuring out which category of advertisements users like and maximizing the CTR click rate [10]. This is the bandit problem. A class of algorithms for solving such repeated decision problems under a model with return uncertainty is called the bandit algorithm. For decision-makers, the only result of a single observation is the reward they get, and it is impossible to observe what kind of reward they will get if they take a different decision. According to different assumptions about the rewarding process, the bandit problem can be mainly divided into three types: Stochastic, Adversarial, and Markovian [9]. The reward of the arm of the Stochastic bandit obeys a certain fixed probability distribution. By contrast, the distribution of the reward of arms of an adversarial bandit will change, but it never equals zero. For Markov-style bandit, the rewards of the arms are defined by Markov chains [11].

The basic algorithms commonly used for MAB problems are $\boldsymbol{\varepsilon}$-greedy, pursuit, reinforcement comparison, Upper confidence bound (UCB), and Thompson sampling (TS). The greedy algorithm chooses the highest confidence at each chosen moment regardless of the action-estimated reward. $\boldsymbol{\varepsilon}$-Greedy is to set an ε to guide how greedy the action will be. For example, set $\boldsymbol{\varepsilon}$ to 0.1 to ensure that 10% of the time is devoted to exploration, and 90% of the time is used for exploitation. The specific operation is to generate a random number from 0 to 1 each time you play. If the number is greater than $\boldsymbol{\varepsilon}$, take the action with the highest estimated probability of winning. If it is less than $\boldsymbol{\varepsilon}$, randomly chose another action (including the action with the highest probability of winning), and after obtaining the profit, update the estimated winning probability of this action for reference in the next election. In the short term, the greedy algorithm is more dominant, but in the long run, proper exploration is more beneficial to the total gain accumulation. The advantage of this method is that it is simple to implement and not too bad, even if the distribution is non-stationary. The disadvantage is also obvious that it usually converges slowly; The ratio of performing the optimal action (greedy) after $\boldsymbol{\varepsilon}$ converges is $\mathbf{1} - \boldsymbol{\varepsilon} < \mathbf{1}$. For this, it may be possible to let $\boldsymbol{\varepsilon}$ decrease over time [12].

Upper confidence bound algorithm is action selection based on the upper confidence bound. The maximum action value computed in this algorithm is an upper bound on the possible true value of the action. If an action is selected, its uncertainty becomes smaller, while if it is not selected, the uncertainty of the action becomes larger. As the total number of choices increases, all actions will be chosen, but actions with lower value estimates or actions that have been chosen more times are chosen less frequently. Actions that are selected less frequently are more likely to be selected first because of the larger upper limit. The core idea of upper confidence bound selection is that the square root is a measure of the accuracy or variance of the action value estimate. In this algorithm, all actions are selected, but actions with low estimates or trivial actions are selected less frequently [13].

TS was first proposed by William R. Thompson in 1933[14]. Thompson-sampling samples from the beta distribution of each action and chooses the action with the highest return value. Such a selection method also allows actions that are not frequently attempted to have a wider range of possible values because they will have a wider distribution due to the properties of the beta distribution. An obvious feature of UCB, a deterministic algorithm, and TS, which is a random algorithm, is that UCB needs to update the upper bound in real-time, while TS allows delayed updates. In practical applications, TS will have a better practical application effect [15]. The TS algorithm is

an implementation of Probability matching because it is sampled from the prior distribution, and the reward probability corresponding to each action is based on the facts that have been observed and the current iteration result.

Multi-armed bandit is a very typical decision-making method, which is widely used in recommender systems. Advertisers need to discover and optimize the creative form of advertising early, which involves the rational allocation of investment and better advertising effect, and ultimately improves the return on investment [16]. While widely used standard A/B tests require a week-long data collection cycle to make a final decision, a multi-arm algorithm can accomplish the same task in less time. Another area of widespread use is in medicine, exemplified by the treatment of multiple myeloma. In 2019, Wang, Yingfei, et al. [17] proposed the use of multi-armed bandit machines to determine the order of treatment with background information about patients and treatments, to maximize overall survival outcomes and used TS to alleviate the possible absence of numerable observations needed in heterogeneity.

In this paper, the contributions are threefold. Firstly, the multi-armed bandit problem is modelled and the classical solutions to it are introduced, including ε-greedy algorithm and UCB algorithm. Secondly, TS algorithm is derived and discussed, which consists in playing an action according to the probability that maximizes the expected reward. Thirdly, cases are shown to demonstrate the applications of TS algorithm in various fields in recent years and the related improvements which expand the scope of application and improve the efficiency of decision learning.

2. Problem formulation

2.1 The multi-armed bandit problem

This paper will start with the stochastic multi-armed bandit problem [18], which assumes that a bandit machine has K arms. Under a limited number of rounds $t = 1,2,3,\cdots$, one arm must be selected each time and a total goal is to maximize a cumulative reward value. Each arm, when selected, generates a reward value following some unknown but fixed distribution. Specifically, each time an arm is selected, it will sample from its corresponding distribution to get the feedback (bounded [0,1]) reward and return it. In addition, setting of the reward follows bandit feedback, that is, only the reward of the selected action can be observed. The combination of all arms of a multi-armed bandit machine can be regarded as a set of real distributions $B = \{R_1, \dots, R_k\}$, and the reward of the k^{th} arm obeys the distribution R_k. Often in the actual process, more consideration is given to the reward mean vector. Definition $\{\mu_1, \dots, \mu_k\}$ is the mean reward associated with these reward distributions, where $\mu_k = E[R_k]$. Since the distributions are unknown, the expectations are also unknown. The optimal mean reward is defined as $\mu^* = max\{\mu_k\}$. The difference between the mean reward of any arm and the mean reward of the best one is expressed as $\Delta_k = \mu^* - \mu_k$. Regret is a globally defined metric that can be used to measure how well an algorithm performs. It is defined as the expected difference between the sum of rewards when the optimal strategy is adopted and that having collected: known as regret at round T. Since choices of arms are randomly generated by an algorithm, $R(T)$ is also a random variable, therefore expected regret is usually considered as an evaluation criterion. Define $N_k(T)$ as the number of times the arm i has been pulled by step $t-1$. Then the expected total regret in round T is defined by

$$E\big(R(T)\big) = E\big(T\mu^* - E(\textstyle\sum_{t=1}^{T}\mu_k)\big), \tag{1}$$

2.2 ε-Greedy Algorithm

ε-greedy uses random exploration, uses currently known information to make decisions, and trades off exploration and utilization based on a probability ε. Known information refers to the expectation of reward brought by choosing action a at time t: the expected benefit that action a has produced so far. For example, if the 10^{th} choice is now made, the previous action 1 has been chosen a total of

3 times, and the average of the three returns is calculated. Each time the agent tries to choose one among N items to recommend to the user, explore with the probability of ε, that is, randomly select a rocker with a uniform probability; use it with the probability of $1-\varepsilon$, that is, select the rocker with the highest current average reward arm, where $\varepsilon \in [0,1]$ The larger ε is, the greater the degree of exploration. For example, the current action is randomly selected from K machines with a probability of 0.3, and the machine with the highest expected return is selected with a probability of 0.7. ε is usually taken as 0.1, it can vary widely according to the situation and preference.

One of the shortcomings of the ε-Greedy algorithm is that, throughout the random exploration process, historical information is not considered, and the algorithm may mistakenly continue to explore actions that are known to have large regret values. To reduce the likelihood of such inefficient exploration, one way is to gradually reduce the value of ε over time; another way is to select machine explorations that are less certain but have potential.

2.3 UCB Algorithm

What Upper Confidence Bound captures is the change in mean confidence. The reason why it is Upper is that we are analysing reward, which is taking the upper bound of the confidence interval as an estimate under a certain degree of confidence. The Hoeffding inequality [19]

$$P\{|\bar{x} - E(\bar{x})| \leq \delta\} \geq 1 - 2e^{-2n\delta^2} \tag{2}$$

Is used to evaluate the upper bound. According to the law of large numbers, with the accumulation of samples, n continues to increase, and the bound will gradually tighten. Therefore, p is defined by a function $p = 1 - 2e^{-2n\delta^2}$ whose form is a Gaussian distribution. If given determining p, δ can be calculated, which is the distance from the supremum to the mean. By taking the inverse of p, the upper confidence bound can be deduced as

$$U_k = \mu_k + \sqrt{\frac{2\ln T}{n}}, \tag{3}$$

where μ_i represents the estimated expected value of arm k, T represents the number of times the test has been tried so far, n represents the number of times the arm k has been shaken down.

The right side of formula (2) is divided into two parts, the left part represents the current estimated value of arm k, and the right side can be understood as the standard deviation of arm k. The part on the right can be seen as a compromise between exploration and exploitation. Because when the number of explorations of arm k is very small, n is very small, then the right equation is large, the value of the entire equation is large, and the probability of arm k being selected is very high, which belongs to the exploration part of arm k. When each arm has been explored enough, the value of n is larger, and the influence of the formula on the right is reduced. Then the value of the entire formula is mainly determined by the estimated value on the left. That is to say, the larger the estimated value is, the higher the probability of arm k being selected, this belongs to the utilization part of arm k. When the number of attempts is sufficient, the estimated value has a greater impact. And the confidence is closer to 1 and the answer is closer to the real expectations of the arm.

One of the shortcomings of the UCB algorithm is that it cannot use context which widely exists in real application scenarios, such as the user ID of the recommender system. Without making full use of the contextual information of the recommended scene, all selection strategies are the same, ignoring the diverse factors and characteristics of specific occasions, such as the user's own interests, preferences, purchasing power and other factors as a living personality. Therefore, the same product is accepted differently by different users and in different situations. If the context information can be used in the system decision-making process, the learning and improvement efficiency of decision-making can be improved and the algorithm can converge to the optimal choice faster. To take

advantage of context, consider learning them using linear regression [20], or estimating the supremum using ridge regression [21].

3. Thompson Sampling

Then, after understanding the relatively early classical algorithms, the problem arises. In recent years, experienced the continuous theoretical research and practical application development, is there an algorithm more natural and natural randomized than the ε-greedy algorithm whose disadvantage lies in the lack of active exploration, and the UCB algorithm, whose core is to use the mean and standard deviation to form an estimate of the item to achieve the effect of E-E leading to the shortcoming of the failure of achieving a truly random effect? Yes, the answer is the Thompson sampling (TS) algorithm. TS algorithm's estimate of the average reward is no longer a sample average but is directly sampled from the current posterior distribution. In this case, the TS algorithm will naturally complete the two tasks of exploration and exploration at the same time. If an arm has not been selected, then the samples sampled from this arm will fall on the entire interval with an approximately uniform probability (equivalent to uniform exploration). And if an arm is selected more times, then the natural estimation is more accurate. If the arm is relatively "good", the probability of sampling from its posterior distribution is relatively high. It is easier to be selected (exploitation).

TS, like UCB, optimizes results by estimating the conversion rate, while TS uses prior knowledge to a greater extent. Bernoulli distribution just has Beta distribution as conjugate prior, with conjugate prior, it is easy to do Bayesian update. Beta distribution [22] is a distribution of random variables on (0, 1). It has two parameters α, β greater than 0. The probability density function is

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}\,du} = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha^{-1}}(1-x)^{\beta-1} = \frac{1}{B(a,\beta)}x^{\alpha-1}(1-x)^{\beta-1}. \quad (4)$$

In Equation 4, the normalizer is a normalization factor, which exists to make the integral of probability 1. In TS, each arm is set with independent Beta distribution, and each arm is sampled once, and then the arm with the largest number is selected. After each recommended arm is selected, prior knowledge of the probability of each arm producing a reward is used to set the parameters of each beta distribution. The beta distribution with the new parameters is a more accurate estimate of the return of the selected arm after incorporating this experience. At time t, the expectation of reward for behavior a is sampled, and the sampling is based on the Beta distribution Beta $(\alpha_k + 1, \beta_k + 1)$, where α_k and β_k correspond to the k^{th} arm from the start to the time point t. Rewards the number of 1s and 0s, the purpose of the action '+1' is to prevent the denominator from being 0 during the calculation. If we know that the conversion rate of an item is 0.25 and the variance is about 0.00186 before the test, then we can get the prior α=25, β=75, and then add 1 to α every time the reward is 1, otherwise β plus 1.

	Algorithm 1 BernTS (K, α, β)	
1	**for** $t = 1,2, \ldots$ **do**	
2	**for** $k = 1, \ldots, K$ **do**	
3	Sample $\hat{\theta}_k \sim beta(\alpha_k, \beta_k)$	
4	**end for**	
5	$k^* \leftarrow \arg max_k \hat{\theta}_k$ Recommend action	▷
6	Apply action s_{k^*} and observe rw_{k^*}	
7	Update $\alpha_{k^*}, \beta_{k^*}$: $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_k + rw_t, \beta_k + 1 - rw_t)$	
8	**end for**	

TS algorithm is outlined as Algorithm 1. $beta(\alpha_k, \beta_k)$ represents a beta distribution with parameters α and β. The algorithm takes input of the number of resources K and the initial parameters of the beta distribution of each resource. At each discrete time step t, the posterior reward for each resource is sampled and the resource with the highest reward is selected (Lines 2-5). The player takes the recommended action, gets the corresponding reward, and uses the reward to update the distribution parameters for the selected action (Lines 6-7). A Bernoulli bandit generates either a zero or a positive unit reward, i.e.,

$$rw_k = \begin{cases} 0, Z = \phi \\ 1, else \end{cases}, \tag{5}$$

TS captures changes in confidence in the estimate of the mean as the data accumulates. However, the disadvantage of classic TS is that it is only suit to be used for the Bernoulli bandit. Some improved TS algorithms [23] have been proposed in recent years, which expand the scope of applications not only limited to Bernoulli bandit, and the application of improved algorithms will also be introduced in Chapter 3.2.

4. Applications of TS

4.1 Advertising

As the sale of advertising space in digital media through real-time ad auctions has become popular over the past decade, multiple auction platform that allow advertisers to bid for advertising space in the auction, also called Supply-Side Platform (SSP), came into being. For an SSP in the advertising world, there are many bidders in the top bid auction that are other SSPs. Grégoire Jauvion et al wrote the optimization of the revenue of an SSP as a contextual bandit problem [24]. Contextual information for each bid contains available information about ad slot opportunities, such as internet user attributes or ad placement, and the closing price of the SSPs internal auction. There is now a need to develop a sequential strategy for bidding to deliver as many ads as possible for as little as possible to web publishers who want to sell ad space. To solve this contextual slot machine problem, a TS algorithm combined with a Bayesian algorithm and a particle filter is used. In each context c, the highest bid X_i among other SSPs in the header auction is modelled as a distribution Φ_c, and the posterior distribution of the discrete approximation is sequentially updated by a Bayesian method using particle filtering with a lognormal distribution.

Assume that S SSPs: $S_1, \ldots, S_s$ compete in the header bidding auctions. Note D_t the sequence of impressions happening before time t (including t), $D_{t,c}$ the subsequence of D_t containing all impressions k such that $c_k = c$. The revenue function $R_k(.)$ of S_1 at impression k can be written as $R_k(q) = 1_{q \geq x_k}(p_k - q)$, where p_k is the amount paid by the advertiser winning the internal auction. Suppose f_θ is a family of distributions parametrized with θ, F_θ is the corresponding cumulative density functions. Φ_c belongs to the family f_θ, so we note θ_c be such that $\Phi_c = f_{\theta_c}$. After fixing a prior distribution $\pi_{c,0}(\theta)$ over θ_c, the posterior distribution $\pi_{c,t}(\theta)$ is considered to

give all the observations available at the end of the t-th auction for all t. From the Bayes rule, $\pi_{c,t}(\theta)$ is defined as:

$$\pi_{c,t}(\theta) \propto \pi_{C,0}(\theta) \prod\nolimits_{k \in D_{t,c}} \left[F_\theta(q_k) 1_{x_k \le q_k} + \left(1 - F_\theta(q_k)\right) 1_{x_k > q_k}\right]. \tag{6}$$

A value θ in context c is sampled from the posterior distribution $\pi_{c_k,t_{k-1}}$ in the k-th experiment. Then calculate the bid $q_{k_{max}}$ that maximizes the SSP's revenue expectation

$$\max_q (p_k - q) F_\theta(q) \tag{7}$$

If $x_k \sim f_\theta$. Finally observe the auction results $1_{q_k \ge x_k}$ and update the posterior probability π_{c_k,t_k}.

Using context information and TS and adding particle learning can effectively improve the efficiency and accuracy of the optimal solution selection strategy. Stochastic modelling relies on the highest bid x_k among other SSPs, and only then introduces a payoff function that takes full advantage of contextual information and is undisturbed. Using particle filter can better maintain more stable performance in data drift compared to UCB or non-stationary bandit algorithm. Even the computational cost and additional parameters added by the introduction of particle filters do not affect their efficiency and selection accuracy too much. If not only consider the independent context but represent the context as a continuous variable, maybe this algorithm can be further improved.

Two different state-of-the-art strategies in the field are used as baseline strategies: UCB [25] and EXP3 [26] to compare TS strategies. For all three strategies, the average reward

$$\frac{1}{n} \sum\nolimits_{k=1}^{n} R_k(q) = \frac{1}{n} \sum\nolimits_{k=1}^{n} 1_{q_k \ge x_k} (p_k - q_k) \tag{8}$$

Is used to measure the performances of the strategies after n auctions.

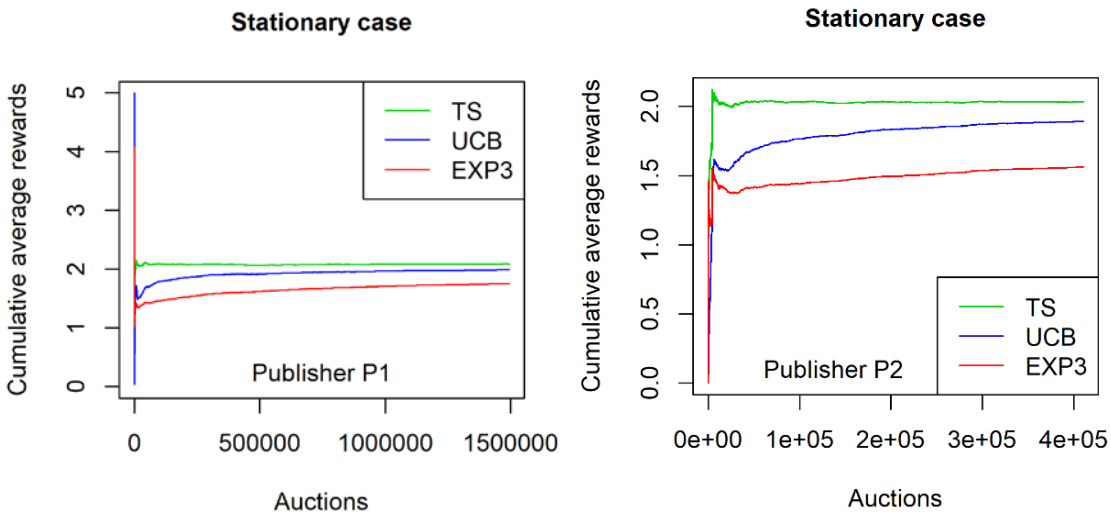


Figure 1. Evolution of the average rewards of TS, UCB, and Exp3 for dataset P_1 and P_2 (stationary environment) [24].

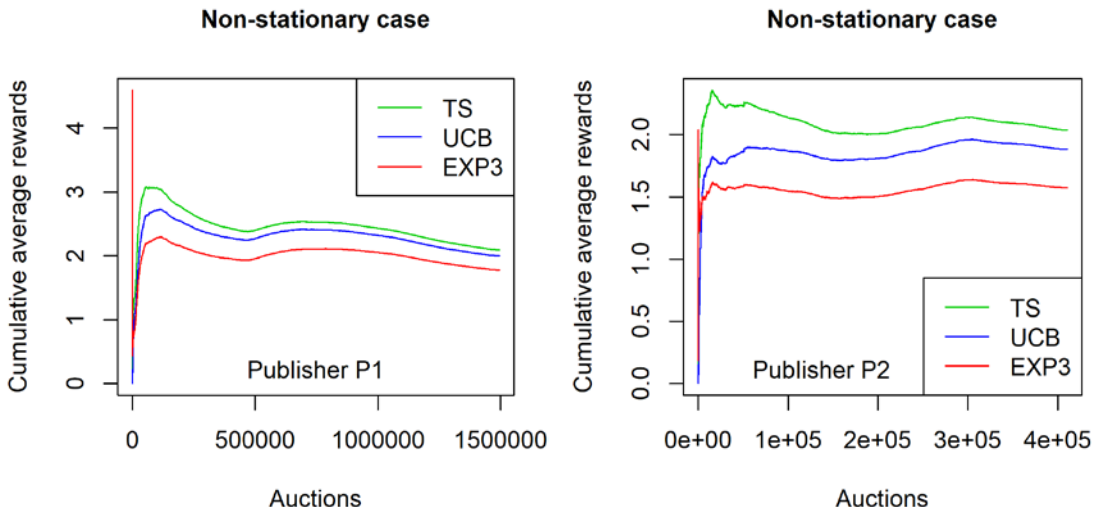


Figure 2. Evolution of the average rewards of TS, UCB, and Exp3 for dataset P_1 and P_2 (non-stationary environment) [24].

Fig.1 depicts the average reward as a function of n on datasets P_1 and P_2 in a stationary environment (i. e. on the shuffled dataset), and Fig.2 depicts a non-stationary environment (i. e. on the ordered dataset). Results on the same dataset are plotted separately. The average rereward of the TS strategy is significantly higher than that of the baseline strategies EXP3 and UCB in both environments, and the convergence rate is expressed as the minimum number of auctions required for the overall average return of the strategy on the entire dataset. The convergence rate of the TS strategy is faster than that of EXP3 and the UCB strategy, which means that using the TS strategy has a greater possibility of using a smaller number of auctions, which improves the efficiency of reward.

On the dataset P_1, the average reward with TS strategy is 2.0888 for the stationary case and 2.0937 for the non-stationary case. The corresponding success rates (i. e. the share of auctions won $n^{-1}\sum_{i=1}^{n} 1_{q_k \geq x_k}$) are 32.19% and 32.09% respectively.

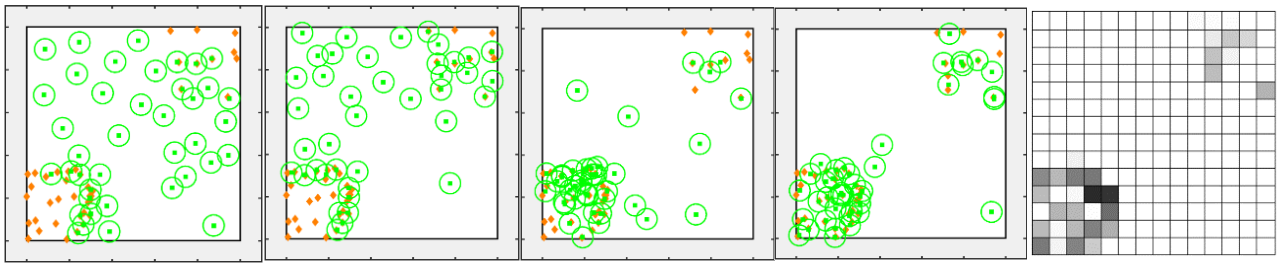
4.2 Multi-Target Searching and Tracking

The multi-target robot search task mainly refers to a plane area that has no prior knowledge of the target distribution before the search. The robot wants to find some targets as quickly as possible through continuous attempts, and finally determine the distribution of all targets. The process of this agent constantly trying and getting rewards for learning fits well with the MAB problem. To make the multi-robot system better cope with the monitoring task of uneven target distribution in the designated area, Chen et al combined Bernoulli TS and Lloyd's algorithm in the distributed control strategy [27]. Robots participating in the search need to choose places that are considered to be easier to obtain rewards to try, and also choose to try (explore) some less-understood areas considering whether the unknown areas have a greater chance of obtaining rewards. In target searching, the rewards are targets and no targets. If through sampling, this area brings rewards, it proves that the target is likely to be densely distributed in this area but cannot be determined. As the time of sampling increases, the value of α increases, and the target distribution density in some areas is constantly believed to be relatively high, that is, it is very likely to find the target and obtain rewards if robots search there. But robots also need to sample areas with low density, because although the reward may not be obtained, in order to prevent the possibility of these areas bringing a higher cumulative reward in the long-term.

Algorithm 2 Distributed Search

for $r_i \in R$ **do** ▷ Initialize robots
  $g_i = q_i$
  for $k \in \{1, \dots, K\}$ **do** ▷ Initialize beta functions
    $\alpha_k \leftarrow 1,\ \beta_k \leftarrow 1$
  end for
end for
for $t = 1,2, \dots$ **do**
  Receive measurement set Z_i
  Find nearest action index $\hat{k} = \arg\min \|q_i - s_k\|$
  Compute rw_k
  Update $\alpha_{k^*}, \beta_{k^*}$
  Broadcast (i, t, k^*, rw_{k_i})
  if $q_i = g_i$ **then** ▷ Reached goal
    if $rw_{k_i} = 0$ **then**
      Select k_i^* using Algorithm 1
      Set goal $g_i = s_{k_i^*}$
    end if
  end if
end for

An active search strategy of the Bernoulli-TS method is mainly used in robots that are not in the group of robots that are actively tracking the target. The entire environment to be developed is evenly divided into K regular polygon areas, and the centre of each regular polygon corresponds to a sampling position s_k, which can be regarded as an arm, then the set of actions is $S = \{s_1, \dots, s_k\}$. After the robot selects an action and has a delay in reaching the target, it gets a reward rw_k (as formula (5)) for that position (in the Bernoulli model reward mechanism) after the exploration task is transformed into a MAB problem, the TS variable of the dynamic reward probability is subjected to beta distribution Update so that the robot can also react to future reward probability changes and continue to optimize exploration in the presence of moving targets in a specific area, and set the robots to broadcast (i, t, k^*, rw) tuples to each other through the communication graph, updating the local α and β copies, Eventually, a shared α and β are achieved. Without prior knowledge, the newly developed distributed control algorithm speeds up the target search, especially when many targets are concentrated in some fixed regions in the environment that are not uniform, that is, the roughness provided by α and β. The global information optimizes task assignment by comparing the number of objects among different small blocks, so that the robot can efficiently search and track multiple moving objects. Related work is also found in [28] [29].



(a) Lloyds 40s (b) Lloyds 60s (c) Lloyds+TS 40s (d) Lloyds+TS 60s (e) Values

Figure 3. Figures show comparison of applying pure Lloyd's algorithm and a combined Lloyd's algorithm with Thompson sampling after 100s. In Fig 3(a)-(d), green squares and circles show robot locations and sensor footprints, respectively. Orange diamonds show the locations of targets. Fig 3(e) maps values in Fig 3(d) by darkness, with a darker colour indicates a higher value [27].

The author compares the new method incorporating the TS algorithm with the previous method [30] through a single experiment of multi-target searching. There are 40 robots searching for 40 objects in environment E, which is assumed as a $100\text{m} \times 100\text{m}$ square. The 40 stationary targets randomly drawn in the two defined $33\text{m} \times 33\text{m}$ sub-areas with 30 targets in the lower-left sub-environment and 10 targets in the upper-right sub-environment respectively.

Figure 3 compares the positions of the robots and the targets at different time points during exploration using the two algorithms. Significantly different from the distribution of robots using only Lloyd's algorithm at various time points in Figure 3(a)-(b), the robot team using the distributed TS strategy rapidly learns the target distribution and the clustering of regions that may contain targets. After that, as shown in Fig. 3(c)-(d), at 40 seconds, a large number of robots have gathered on the large cluster of the target, and the remaining few are exploring, and at 60 seconds, most robots have found the target, And the distribution of the number of robots closely matches that of targets (lower-left: upper-right is 28:9), when 3 robots continue to monitor in other unlikely regions, indicating that individuals are more likely to be assigned to areas that require more individuals to allow team to search and track targets faster. Fig. 3(e) reflects the value of α for each sampling candidates after 60 seconds, implying that the team received more reward in regions with higher target concentrations.

4.3 Fog Computing

Multi-interface channel allocation decision problem of fog computing has been discussed by Junge Zhu et al in 2021 with the idea of multi-play multi-armed bandit and use TS to learn binary transmission feedback, aiming at actively make online channel allocation decisions, reduce performance loss, and finally develop a multi-interface channel allocation with binary feedback (MICA-B), which is successfully validated as an effective fog computing integrated design [31]. Each fog node is regarded as an agent. In the total time slot T, each fog node maintains a fixed set C consisting of N independent channels. The available channel set $C = \{1, \cdots, N\}$ is regarded as an arm set. And according to the working mechanism that M channels is allocated to the interfaces at each time slot t, each subset with M channels is denoted as an arm K_t. At the end of each time slot, the unknown distribution binary transmission feedback message $\{X_k(t)\}_{k \in K_t}$ is received by the fog nodes as a reward. The goal to be maximized in the fog computing channel allocation scheme is the total expected throughput

$$\max_{[K_t^\pi]_t} E\left[\Sigma_{t=1}^{T} \sum_{k \in K_t^\pi} r_k X_k(t)\right] \tag{9}$$

generated by successful transmission. Because the success probability of all channels is difficult to be used as prior information in practice, TS is introduced to combine the decision process and the combination of online learning, dealing with exploration and development problems, reduces the possible performance loss under uncertainty. An initial α and β are derived from empirical information

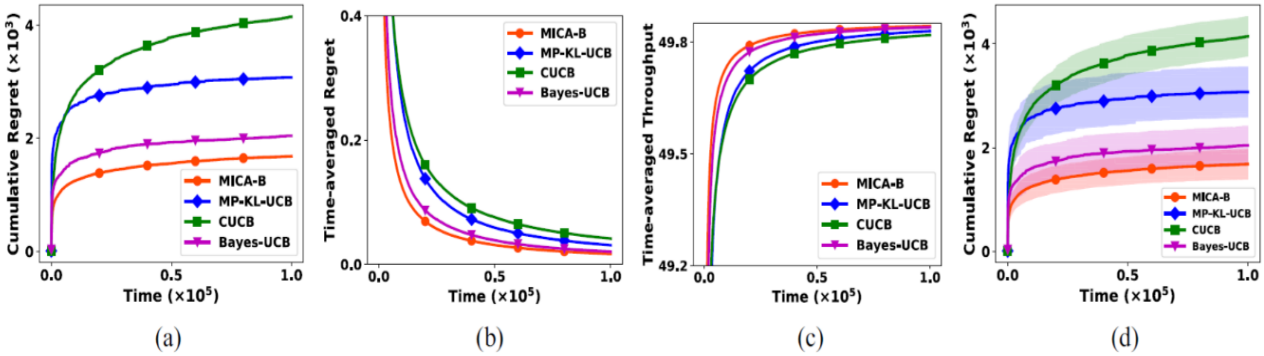


Figure 4. Comparison of MICA-B and the baselines under the high-variance setting in the binary-feedback scenario. (a) Cumulative regret. (b) Time-averaged regret. (c) Time-averaged throughput. (d) Variance [30].

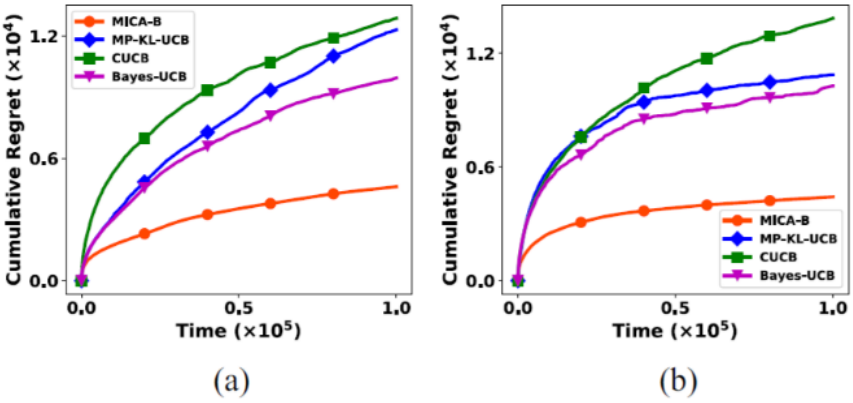


Figure 5. Comparison of regret under the medium-variance and low-variance settings in the binary-feedback scenario. (a) Medium-variance. (b) Low-variance [31].

such as transmission logs, the transmission success probability of each channel is estimated with a normal distribution, and the subset of channels with the largest total estimated value is selected. Then, the parameters of this distribution are continuously updated according to the feedback signal x_k. Another advantage of this approach is that if the transfer rate is uniform, the total regret over time is guaranteed to be within an upper bound.

Three commonly adopted strategies in the field are used as baseline strategies: CUCB [32], MP-KL-UCB [33] and Bayes-UCB [34] to compare the MICA-B strategy. The regret defined in this case is the gap between the expected and optimal cumulative throughput produced by the policy

$$R_\pi(T) \triangleq \sum_{t=1}^{T} \sum_{j=k^*} r_j p_j - E\left[\Sigma_{t=1}^{T} \sum_{k \in K_t^\pi} r_k X_k(t)\right], \tag{10}$$

Where less regret means higher throughput.

Fig. 4 compares the variance of cumulative regret, time-averaged regret, time-averaged throughput, and cumulative regret in the high-variance setting in the binary feedback context. From Fig. 4(a)-(c) and Fig. 5(a)-(b), in the same scenario with high, medium and low variance, MICA-B improves the learning efficiency due to the use of prior information, with lower regret and higher throughput, respectively. It can be seen from Fig. 3(d) that the variance of accumulated regret for MICA-B is also the smallest, which means that this strategy contributes to more stable wireless transmission.

4.4 Discussions

The multi-armed bandit problem has found its own application scenarios in current industrial applications, especially in online recommendation systems, where data is easy to obtain, and automation is the only way to expand, such as in real-time online recommendation systems, search sorting, etc. The system is widely used. Usually, the agent wants to do exploration, know which

choice has the best behavior, and wants to maximize the cumulative reward in the shortest time, so TS is the solution based on this. In the UCB algorithm that improves the inefficient random exploration of ε-greedy, UCB explicitly appears in the item of regret, including the value of UCB that the algorithm needs to use directly. In the TS algorithm, the algorithm does not need to use the values of UCB and LCB directly, but artificially introducing UCB and LCB as the Bayesian regret in the analysis will make the analysis more effective. The application prospects of TS are actually very, very broad, ranging from choosing a route to go to work, to making a fortune in a casino stud. At present, the more popular research applications are the task assignment and target search problems in machine learning, and the cold start problem in recommender systems, aiming to study and improve the accuracy of pushing advertisements to users.

However, TS also has some shortcomings. First, when the posterior distribution is very complex, the convergence speed of TS will be very slow. Second, when the sampled items have close click-through rates, these items are computed and sampled all the time, which can be time-consuming, and third, in most practical applications, the posterior distribution is maintained and sampled on the model is computationally tedious. For the first point, maybe try a more advanced sampling method Importance Sampling [35] or MCMC [36]. For the third point, TS is often used in conjunction with approximate sampling techniques. In view of the limitations of the classic TS algorithm, how to further expand the distribution range of the bandit machine application is not only applicable to the Bernoulli bandit machine, but also to the vast majority of probability distributions is also an important aspect in the future. If it can be universal to more distributions, efficient TS can also have wider application prospects, not only in recommender systems, using TS or as a prerequisite for building certain learning types. The machine learning optimization of the conclusion can also achieve and continue to improve speed.

5. Conclusions

This paper formalizes the multi-armed bandit problem and defines metrics to measure the quality of the algorithm: reward and regret. Any question about choice can be transformed into a multi-armed bandit problem. Their purpose is to maximize the profit, and the best way is to try it strategically as soon as possible. These strategies are the MAB algorithm. This paper mainly focuses on three commonly used algorithms: Epsilon-Greedy algorithm, Upper Confidence Bound algorithm and TS algorithm. The Epsilon-Greedy algorithm is characterized by defaulting to the currently known handle with the highest return, and occasionally selecting those that do not have the highest return. Not caring how many times each handle has been pulled means that these algorithms will no longer select handles with particularly low initial returns. The UCB algorithm not only pays attention to the return, but also pays attention to the number of times each handle is explored. UCB adopts a deterministic selection strategy and uses a probability distribution (only the upper bound of the confidence interval) to quantify uncertainty, which may result in the same return each time, while TS is a randomization strategy. UCB is more computationally intensive, and TS is relatively simpler to implement. Combining the advantages of the above algorithms, the most popular in recent years is the TS algorithm.

At present, TS is the most computationally efficient and most accurate algorithm is mainly used for initial information, such as the cold start problem where the prior distribution information of the target distribution is missing. In the future, for the optimization of the algorithm, regardless of the substantial increase in computing power, the feasible direction is still to expand the types of slot machines with different distributions that the TS algorithm can be applied to and use in combination with other algorithms to improve computing efficiency and increase the number of simultaneous processing. the number of candidate arms without significantly increasing the computational cost.

References

[1] Dewey, D. (2011) Learning what to value. In International Conference on Artificial General Intelligence. Springer, Berlin, Heidelberg. pp. 309 - 314.

[2] Young, H. P. (2009) Learning by trial and error. Games and economic behavior, 65 (2): 626 - 643.

[3] Silver, D., Singh, S., Precup, D., Sutton, R. S. (2021) Reward is enough. Artificial Intelligence, 299: 103535.

[4] Odell, J. J., Parunak, H., Fleischer, M., Brueckner, S. (2002) Modeling agents and their environment. In International Workshop on Agent-Oriented Software Engineering. Springer, Berlin, Heidelberg. pp. 16 - 31.

[5] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. (2013) Playing atari with deep reinforcement learning. arXiv preprint arXiv: 1312.5602.

[6] Fu, M. C. (2019) Simulation-based algorithms for Markov decision processes: Monte Carlo tree search from AlphaGo to Alpha Zero. Asia-Pacific Journal of Operational Research, 36 (06): 1940009.

[7] Berger-Tal, O., Nathan, J., Meron, E., Saltz, D. (2014) The exploration-exploitation dilemma: a multidisciplinary framework. PloS one, 9 (4): e95693.

[8] Berry, D. A., Fristedt, B. (1985) Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability). London: Chapman and Hall, 5 (71-87): 7 - 7.

[9] Bubeck, S., Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. arXiv preprint arXiv: 1204.5721.

[10] Lu, T., Pál, D., Pál, M. (2010). Contextual multi-armed bandits. In Proceedings of the Thirteenth international conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings. pp. 485 - 492.

[11] Zheng, R., Hua, C. (2016) Sequential learning and decision-making in wireless resource management. Springer International Publishing.

[12] Watkins, C. J. C. H. (1989) Learning from delayed rewards.

[13] Lai, T. L., & Robbins, H. (1985) Asymptotically efficient adaptive allocation rules. Advances in applied mathematics, 6 (1): 4 - 22.

[14] Thompson, W. R. (1933) On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika, 25 (3-4): 285 - 294.

[15] Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., Wen, Z. (2018) A tutorial on TS. Foundations and Trends® in Machine Learning, 11 (1): 1 - 96.

[16] Schwartz, E. M., Bradlow, E. T., Fader, P. S. (2017) Customer acquisition via display advertising using multi-armed bandit experiments. Marketing Science, 36 (4): 500 - 522.

[17] Wang, Y., Zhou, Z., Mamani, H., Coffey, D. G. (2019) How do tumor cytogenetics inform cancer treatments? dynamic risk stratification and precision medicine using multi-armed bandits. Dynamic Risk Stratification and Precision Medicine Using Multi-Armed Bandits (June 17, 2019).

[18] Bubeck, S., Munos, R., Stoltz, G. (2009) Pure exploration in multi-armed bandits problems. In International conference on Algorithmic learning theory. Springer, Berlin, Heidelberg. pp. 23 - 37.

[19] Hoeffding, W. (1994) Probability inequalities for sums of bounded random variables. In The collected works of Wassily Hoeffding. Springer, New York, NY. pp. 409 - 426.

[20] Foster, D., Rakhlin, A. (2020) Beyond ucb: Optimal and efficient contextual bandits with regression oracles. In International Conference on Machine Learning. PMLR. pp. 3199 - 3210.

[21] Chu, W., Li, L., Reyzin, L., Schapire, R. (2011) Contextual bandits with linear payoff functions. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings. pp. 208 - 214.

[22] Gupta, A. K., Nadarajah, S. (2004) Handbook of beta distribution and its applications. CRC press.

[23] Gupta, N., Granmo, O. C., Agrawala, A. (2011) TS for dynamic multi-armed bandits. In 2011 10th International Conference on Machine Learning and Applications and Workshops (Vol. 1). IEEE. pp. 484 - 489.

[24] Jauvion, G., Grislain, N., Dkengne Sielenou, P., Garivier, A., Gerchinovitz, S. (2018) Optimization of a ssp's header bidding strategy using TS. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 425 – 432.

[25] Auer, P., Cesa-Bianchi, N., Fischer, P. (2002) Finite-time analysis of the multiarmed bandit problem. Machine learning, 47 (2): 235 - 256.

[26] Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R. E. (2002). The no stochastic multiarmed bandit problem. SIAM journal on computing, 32 (1): 48 - 77.

[27] Chen, J., Dames, P. Active Multi-Target Search Using Distributed TS.

[28] Chen, J., Park. S (2022) Bernoulli TS-based Target Search Algorithm for Mobile Robots.

[29] Chen, J., Xie, Z., Dames, P. (2022) The semantic PHD filter for multi-class target tracking: From theory to practice. Robotics and Autonomous Systems, 149, 103947.

[30] Dames, P. M. (2020) Distributed multi-target search and tracking using the PHD filter. Autonomous robots, 44 (3): 673 - 689.

[31] Zhu, J., Huang, X., Gao, X., Shao, Z., Yang, Y. (2021) Multi-interface channel allocation in fog computing systems using TS. IEEE Internet of Things Journal, 8 (17): 13542 - 13554.

[32] Chen, W., Wang, Y., Yuan, Y. (2013) Combinatorial multi-armed bandit: General framework and applications. In International conference on machine learning. PMLR. pp. 151 - 159.

[33] Tanczos, E., Nowak, R., Mankoff, B. (2017) A kl-lucb algorithm for large-scale crowdsourcing. Advances in Neural Information Processing Systems, 30.

[34] Kaufmann, E., Cappé, O., Garivier, A. (2012) On Bayesian upper confidence bounds for bandit problems. In Artificial intelligence and statistics. PMLR. pp. 592 - 600.

[35] Tokdar, S. T., Kass, R. E. (2010). Importance sampling: a review. Wiley Interdisciplinary Reviews: Computational Statistics, 2 (1): 54 - 60.

[36] Andrieu, C., De Freitas, N., Doucet, A., Jordan, M. I. (2003) An introduction to MCMC for machine learning. Machine learning, 50 (1): 5 - 43.