A Multiarmed Bandit Incentive Mechanism for Crowdsourcing Task Allocation in cellular Network

Michael Shell, *Member, IEEE,* John Doe, *Fellow, OSA,* and Jane Doe, *Life Fellow, IEEE*

Abstract—The multi-armed bandit problem has attracted remarkable attention in the machine learning community and many efficient algorithms have been proposed to handle the so-called exploitation-exploration dilemma in various bandit setups. At the same time, significantly less effort has been devoted to adapting bandit algorithms to particular architectures, such as sensor networks, multi-core machines, or peer-to-peer (P2P) environments, which could potentially speed up their convergence.

Index Terms—Task Allocation, multiarmed bandits, mobile social network, reward maximization

1 Introduction

In a bandit problem, there are m armed-bandits, every time you pull the arm, you will get a reward, while the reward is random. We can you formulate this problem as following.

1.1 Single-Agent Multi-Armed Bandit

Single-agent multi-armed bandit (SA-MAB) problem was first introduced in [1]. In the most seminal setting, the problem models an agent facing the challenge of sequentially selecting an arm from a set of arms in order to receive an a priori unknown reward, drawn from some unknown reward generating process. As a result of lack of prior information, at each trial, the player may choose some inferior arm in terms of reward, yielding some regret that is quantified by the difference between the reward that would have been achieved had the player selected the best arm and the actual achieved reward. In such an unknown setting, the player decides which arm to pull in a sequence of trials so that its accumulated regret over the game horizon is minimized. This problem is an instance of exploration-exploitation dilemma, i.e., the tradeoff between taking actions that yield immediate large rewards on the one hand and taking actions that might result in larger reward only in future, for instance activating an inferior arm only to acquire information, on the other hand. Therefore, SAMAB is mainly concerned with a sequential online decision making problem in an unknown environment. A solution of a bandit problem is thus a decision making strategy called policy or allocation rule, which determines which arm should be played at successive rounds. Policies are in general evaluated based on their regret or discounted reward performance.

As mentioned before, MAB benefits from a wide range of variations in the setting. In the event that arms have different states, the reward depends on arms' states, and the bandit model is referred to as stateful (Markovian). Otherwise, the model is stateless, which itself can be divided to few subsets. As stateful and stateless bandits are inherently different, we discuss them separately in the following.

- *M. Shell is with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332. E-mail: see http://www.michaelshell.org/contact.html*
- *J. Doe and J. Doe are with Anonymous University.*

Manuscript received April 19, 2005; revised September 17, 2014.

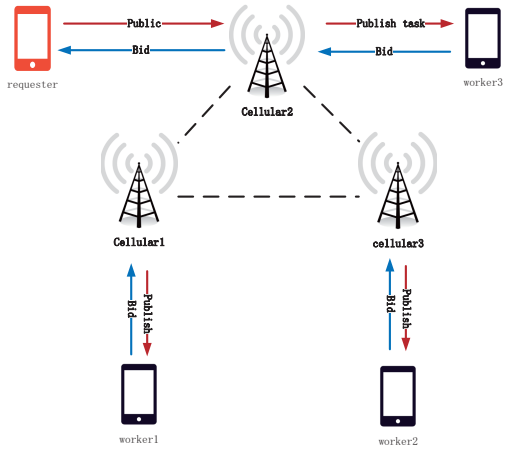

Fig. 1. mobile social network

2 A Parallel with Crowdsourcing

"Crowdsourcing is the act of taking job traditionally performed by a designated agent(usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call", this is what Jeff Howe ever offered the definition of crowdsourcing. The designated agent is often called the requester, while people working on the task are called crowd workers. The requester needs to assign the task to these workers with unknown capability and willingness to complete the task based on the goals to achieve. In mobile crowdsensing network, there is a task publisher at some time, and he/she can chooses a worker from his/her contacts. The task publisher just like the requester, while his/her contacts just like the group of works. Table 1 precisely demonstrates this paralle between crowdsoucing and mobile crowdsensing.

3 Related Work on MAB Problem

4 Model&Problem

4.1 Model for Task Allocation

Consider a network denoted by a graph $G^t = (V^t, E^t)$ at time t with a set $V^t = \{v_i | i = 1, ..., K\}$ of K nodes represented the node set who meet the task requester node represented by v_0 at time t, a set E of edges denoting v_0 can communicate with other nodes. And the graph is star network structure. The system is been divided into time slot, and at each round t, node v_0 has K choices of relay node, where relay nodes are i.i.d. Suppose the ability of every node accomplish a task as stochastic process $\chi_i(t)$ over time with a mean $\mu_i \in [0, 1]$, as done in many previous works [1] [2] [3].

At each round t, an K-dimensional *strategy* vector $\Phi_0(t) = \{\Phi_{0,i} | i = 1, ..., K\}$ is selected under some *policy* from the *feasible strategy set* F. Here $\Phi_{0,i}$ is the index of relay node selected by node carried message in strategy Φ_0. We use i=1,...,K to index strategies of feasible set F in XXXXX. By feasible we mean at that time slot, carried message node meet another k nodes(who can communicate with the message carrier directly). When a strategy Φ_0 is determined, then one node v_i will be selected, and it will give back a reward $r_i(t)$. And then the total reward of T round is defined as $R = \sum_{t=1}^{T} r(t)$. The goal of this allocation problem is to maximize the accumulated reward R over the T rounds. Obviously, if the reward of every node is constant, then we can simply choose the node who will have maximize reward every time slot. While in our problem, the reward is variable over time. And it's probability distribution follows $f(x, \theta)$ where θ is an unknown parameter. The gap between the actual total reward we get and the optimal total reward theoretically is call regret of the algorithm, and it can be formally written as $RE = \sum_{t=1}^{T} r^*(t) - \sum_{t=1}^{T} r(t)$, where r^* is the max reward we can get every time slot. The goal of the algorithm is to keep the regret RE as low as possible.

An allocation strategy(also called a policy) π specifies the action chosen by the task requester(carried message node) at each time in the system. Formally, π is a sequence of random variables $\pi = \{\pi^1, \pi^2, ..., \pi^T\}$ where $\pi^t = \{a_1^t, a_2^t, ..., a_k^t\}$ is the vector of zeros and ones(deterministic mechanism). If $a_i^t = 1$ then node i will be chosen as a relay node at slot t, whereas, $a_i^t = 0$ means node i doesn't be chosen at all, i.e., $\forall t, \sum_i a^t \leq 1$.

The regret of strategy π at time t is defined as follows:

$$RE^{\pi} = \sum_{t=1}^{T} \{r^*(t) - \sum_{i=1}^{K} a_i r_i(t)\} \tag{1}$$

we now define some of the desirable properties that an allocation policy show satisfy:

Definition 1 *Allocation Efficiency(AE)*: An allocation policy π is said to be allocation efficiency if it minimize the regret. Formally, π is allocation efficiency if:

$$\pi = \arg\min_{\pi} RE^{\pi}$$

Definition 2 *Dominant Strategy Incentive Compatibility(DSIC):* Mechanism M is said to be DSIC if truth telling is a dominant strategy for all the workers. Formally,

4.2 Problem

Consider a mobile crowdsensing in the mobile social network. Without loss of generality, we let the requester be user v_0, and suppose that the requester has m homogeneous mobile sensing tasks in one cycle, denoted by $j(n)$. The length of one cycle is T, and these tasks might be different types of task in different cycle. For simplicity, we assume that every requester will choose one worker from who ever get contacted with the requester for every task allocation and the requester's contact is built at the beginning within time t. It means if someone get contact with the requester during the period of the beginning t, the one will be add to the requester's contacts, regardless of who is the caller.

In this paper, we focus on how to choose a sequence of workers so as to maximize the utility of requester in every cycle.

5 Task allocation algorithm

5.1 ϵ-greedy-TA

Algorithm 1 ε-Greedy-TA

Input: K: the number of arms;
R: reward function;
T: the round of times;
ϵ: probability
Output: r:the accumulate reward.

$r = 0$;
$\forall i = 1, 2, ..., K : Q(i) = 0, count(i) = 0$;
for $t = 1, 2, ..., T$ **do**
  if $rand() < \epsilon$ **then**
    randomly choose one from 1,2,...,K and assignment to k
  else
    $k = \arg\max_i Q(i)$
  $v = R(k)$;
  $r = r + v$;
  $Q(k) = \frac{Q(k)*count(k)+v}{count(k)+1}$;
  $count(k) = count(k) + 1$;

if the reward of the arm is very uncertainty, for example, its probability distrubutions contains too much value, then it will need more exploration, and then it need big ϵ value.Vice versa.

5.2 softmax-TA

In softmax algorithm, it acccording to the known mean reward of the arms to decide to choose which arm. If the mean reward is bigger, and the probability of choose this arm is bigger.

And the probability distribution of choose a arm is Boltzmann distributions:

$$P(k) = \frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^{K} e^{\frac{Q(i)}{\tau}}} \tag{2}$$

Q(i) represent the mean reward upon now; $\tau > 0$ is called "temperature", and when τ is very small, the mean reward bigger one will have higher probability to be choose.

Algorithm 2 softmax-TA

Input: K: the number of arms;
R: reward function;
T: the round of times;
τ: temperature parameter
Output: r:the accumulate reward.

$r = 0$;
$\forall i = 1, 2, ..., K : Q(i) = 0, count(i) = 0$;
for $t = 1, 2, ..., T$ **do**
  randomly choose an arm from 1,2,...,K according to equation (1);
  $v = R(k)$;
  $r = r + v$;
  $Q(k) = \frac{Q(k)*count(k)+v}{count(k)+1}$;
  $count(k) = count(k) + 1$;

5.3 Thompson Sampling for Task Allocation

In this section, we will explain our Thompson Sampling algorithm for task allocation with incentive mechanism. The most important challenge is to balance exploitation with exploration. That is, we have two somewhat conflicting goals: (a) quickly find the best arm to pull and (b) pull the best arm as often as possible. Now we'll introduce the asymptotic optimal algorithm: Thompson Sampling.

Before I introduce the Thompson sampling, let's build up some intuition. If we have an infinite amount of pulls, then we know exactly what the expected rewards are, and there is no reason to ever explore. When we have a finite number of pulls, then we have to explore and the reason is that we are uncertain of what is the best arm. The right machinery to quantify uncertainty is the probability distribution. In Bayesian thinking, we want to use the entire probability distribution. Let's define $p_a(r)$ is the probability distribution from which rewards are drawn. That's what controls the bandit. We don't need to estimate the entire distribution for infinite amount of pulls, because all we care about is its mean μ_a. What we will do is encode our uncertainty of μ_a in the probability distribution $p(\mu_a|data_a)$, and then use that probability distribution to decide when to explore and when to exploit.

let's say more clearly about the two probability:
$p_a(r)$ - the probability distribution that bandit a uses to generate rewards when it is pulled.
$p(\mu_a|data_a)$ - the probability distribution of where we think the mean of $p_a(r)$ after observing some data.

With Thompson sampling you keep around a probability distribution $p(\mu_a|data_a)$ that encodes your belief about where the expected reward μ_a is for arm a. For the simple coin-flip case, we can use the convenient Beta-Bernoulli model, and the distribution at round t after seeing S_a, t successes and F_a, t failures for arm a is simply:
$p(\mu_a|data_a) = Beta(S_a, t + 1, F_a, t + 1)$,
where the added 1's are convenient priors.

In our problem, we use Gaussian likelihood function and Gaussian prior to design our version of Thompson Sampling algorithm. More precisely, suppose that the likelihood of reward $r_i(t)$ at time t, given context $b_i(t)$ and parameter μ, were given by the pdf of Gaussian distribution $N(b_i(t)^T\mu, v^2)$. Here, $v = R\sqrt{\frac{24}{\varepsilon}d\ln(\frac{1}{\delta})}$, with $\varepsilon \in (0, 1)$ which parameterizes our algorithm. Let

$$B(t) = I_d + \sum_{\tau=1}^{t-1} b_{a(t)}(\tau)b_{a(t)}(\tau)^T$$

$$\hat{\mu}(t) = B(t)^{-1}(\sum_{\tau=1}^{t-1} b_{a(t)}(\tau)r_{a(t)}(\tau))$$

Then, if the prior for μ at time t is given by $N(\hat{\mu}(t), v^2B(t)^{-1})$, it is easy to compute the posterior distribution at time t+1,

$$\Pr(\hat{\mu}|r_i(t)) \propto \Pr(r_i(t)|\hat{\mu}) \Pr(\hat{\mu})$$

as $N(\hat{\mu}(t), v^2B(t)^{-1})$ (detail of this computation are in Appendix A.1). In our Thompson Sampling algorithm, at every time step t, we will simply generate a sample $\hat{\mu}(t)$ from $N(\hat{\mu}(t), v^2B(t)^{-1})$, and play the arm i that maximizes $b_i(t)^T\hat{\mu}(t)$.

6 Analysis of AT-algorithm

7 Simulation Experiments

8 Conclusion

The conclusion goes here.

Algorithm 3 ThompsonSampling-TA

Set $B = I_d, \hat{\mu} = 0_d, f = 0_d$
for $t = 1, 2, ...,$ **do**
Sample $\hat{\mu}(t)$ from distribution $N(\hat{\mu}(t), v^2 B(t)^{-1})$.
Play arm $a(t) := argmax_i b_i(t)^T \hat{\mu}(t)$, and observe reward r_t.
Update $B = B + b_{a(t)}(t) b_{a(t)}(t)^T, f = f + b_{a(t)}(t) r_t, \hat{\mu}(t) = B^{-1} f$.
if $r_t = 1$ **then**
current arm is i(t), then we need to give him payback.
choose arm $j(t) := \arg\max_{j \in K \setminus i} \theta_j(t)$ and observe reward $r_t{}'$
Perform a Bernoulli trial with success probability $\tilde{r}'_t$ and observe output $r_t{}'$
if $r_t{}' = 1$ **then**
$p_i{}^t = min(p_j{}^t, p_m ax)$
else
$p_i{}^t = min(p_i{}^t, p_m ax)$
else
do not allocation this task at this round.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgments

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.