Stochastic Multi-path Routing Problem with Non-stationary Rewards: Building PayU's Dynamic Routing

Pankaj Trivedi
PayU Payments Pvt. Ltd.
pankaj.trivedi@payu.in

Arvind Singh
PayU Payments Pvt. Ltd.
arvind.singh@payu.in

ABSTRACT

Payment transaction engine at PayU processes multimillion transactions every day through multiple payment gateways. Routing a transaction through an appropriate payment gateway is crucial to the engine for optimizing the availability and cost. The problem is that every transaction needs to choose one of K available payment gateways characterized by an unknown probability reward distribution. The reward for a gateway is a combination of its health and cost factors. The reward for a gateway is only realized when transaction is processed by the gateway i.e. by its success or failure. The objective of dynamic routing is to maximize the cumulative expected rewards over some given horizon of transactions' life. To do this, the dynamic switching system needs to acquire information about gateways (exploration) while simultaneously optimizing immediate rewards by selecting the best gateway at the moment (exploitation); the price paid due to this trade off is referred to as the regret. The main objective is to minimize the regret and maximize the rewards. The basic idea is to choose a gateway according to its probability of being the best gateway.

The routing problem is a direct formulation of reinforcement learning (RL) problem. In an RL problem, an agent interacts with a dynamic, stochastic, and incompletely known environment, with the goal of finding an action-selection strategy, or policy, that optimizes some long-term performance measure. Thompson Sampling algorithm has experimentally been shown to be close to optimal.

KEYWORDS

Machine Learning, Reinforcement Learning, Routing Problem

ACM Reference Format:

Pankaj Trivedi and Arvind Singh. 2018. Stochastic Multi-path Routing Problem with Non-stationary Rewards: Building PayU's Dynamic Routing. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23-27, 2018, Lyon, France*. ACM, New York, NY, USA, Article 4, 6 pages. <https://doi.org/10.1145/3184558.3191630>

1 INTRODUCTION

PayU is a payment gateway aggregator which integrates with multiple payment gateways capable of processing online card payments. PayU also integrates merchants who are selling their products and services online. Each payment gateway integrated with PayU

platform has an unknown probability of making a successful transaction at a given point in time (Figure 1). It becomes very critical for the business to be able to route a transaction to appropriate payment gateway so that the probability of a successful transaction is maximized. This may be modeled as a routing problem wherein an appropriate payment gateway must be selected for each transaction so that overall success rate is maximized. Every payment gateway also has a known but different cost to process the successful transaction. The cost factor to each gateway adds another dimension of optimization.

It is intuitive to apply reinforcement learning technique to learn from the feedback supplied in form of results of processed transactions. The learning is further used to recommend the appropriate gateway for each transaction.

2 RELATED WORK AND MODIFICATION

The approach in this paper is a modification of multi-armed bandit problem. In a multi-armed bandit (MAB) problem a gambler needs to choose at each round of play one of K arms, each characterized by an unknown reward distribution. Reward realizations are only observed when an arm is selected, and the gambler's objective is to maximize his cumulative expected earnings over some given horizon of play T . Since their inception, MAB problems with various modifications have been studied extensively in Statistics, Economics, Operations Research, and Computer Science, and are used to model a plethora of dynamic optimization problems under uncertainty; examples include clinical trials ([3]), strategic pricing ([17]), investment in innovation ([18]), packet routing ([19]), on-line auctions ([7]), assortment selection ([20]), and online advertising ([21]), to name but a few.

In this paper, we focus on a MAB formulation which allows for a broad range of temporal uncertainties in the rewards, while still maintaining mathematical tractability. The uncertainties in the rewards are introduced due to various reasons of a transaction failure. The other variation in the formulation is introduced in the decaying rewards because the exploitation is required to be biased towards recent history. The transaction routing is also required to be cost effective and hence another modification is required to consider cost variation of various gateways. The business requires various knobs to control the tradeoff between success rates and cost. The above mentioned requirements have been accomplished by significantly modified techniques.

3 PROBLEM FORMULATION

Let $\chi = \{1, \dots, K\}$ be a set of gateways. Let $\Gamma = \{1, 2, \dots, T\}$ denote the sequence of decision epochs faced by the recommendation system. At any epoch $t \in \Gamma$, a recommender selects one of the K gateways. When a transaction is processed through the gateway

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW '18 Companion, April 23-27, 2018, Lyon, France

© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.

ACM ISBN 978-1-4503-5640-4/18/04.

<https://doi.org/10.1145/3184558.3191630>

Figure 1: Instances of variation in the expected rewards of various gateways

$k \in \chi$ at epoch $t \in \Gamma$, a reward $X_k^t \in [0, 1]$ is obtained depending on the result of transaction processing, where X_k^t is a random variable with expectation

$$\mu_k^t = E[X_k^t] \quad (1)$$

We denote the best possible expected reward at decision epoch t by μ_t^*

$$\mu_t^* = \max_{k \in \chi} \mu_k^t \quad (2)$$

The goal is to maximize the expected total reward in time T . It is more convenient to work with the equivalent measure of expected total regret: the amount we lose because of not playing optimal gateway in each step. Then the expected total regret in time T is given by

$$E[R(T)] = E \left[\sum_{t=1}^T (\mu_t^* - \mu_k^t) \right] \quad (3)$$

4 SOLUTION

4.1 Reinforcement Learning

Reinforcement learning is a class of methods whereby the problem to be solved by the control system is defined in terms of payoffs (which represent rewards or punishments). The aim of the system is to maximize the payoffs received over time. Therefore high payoffs are given for desirable behavior and low payoffs for undesirable behavior. The system is otherwise unconstrained in its sequence of actions referred to as its policy used to maximize the payoffs received. In effect, the system must find its own method of solving the given task.

A block diagram of a reinforcement system is shown in Figure 2, which shows the basic interaction between a controller and its environment. The payoff function is fixed, as are the sensors and actuators which really form part of the environment as far as the control system is concerned. The control system is the adaptive part which learns to produce the control action a in response to the state input x based on maximizing the payoff r .

Figure 2: Reinforcement Learning System.

The algorithm starts in an ignorant state, where it knows nothing, and begins to acquire data by testing the system. As it acquires data and results, it learns what the best and worst behaviours are (in this case, it learns which gateway is the best).

Environment

In our transaction routing problem, the environment is the current probable approval rates of various gateways, which are unknown at any given point in time. The environment is hence non-stationary and non-stationary environment implies that the old information is potentially less relevant due to possible changes in the underlying rewards. For example, if a payment gateway was processing transactions at an approval rate of 80 percent two hours ago, the gateway may currently be down with zero approval rate.

Reward

Every transaction, once tried with one of the eligible gateways, results into a success or a failure which translates into the reward using the payoff function.

Recommender

The control system acts as a recommender for every transaction. The recommender takes a set of eligible gateways as input and responds with the best gateway to process the transaction depending on the state of environment it has learnt so far.

Feedback

The reward obtained after processing the transaction is fed into the control system to obtain the new state of the environment.

Any transaction initiated in the system would ask for recommendation for payment gateway, and then the transaction is tried with the recommended gateway, the result of the trial is fed into the learning algorithm as reward, for example if the transaction was successful, it is considered 1 as reward for the gateway and if it

is failed the reward is 0. This arrangement is called a Bernoulli arrangement as the rewards are only 0 and 1. The learning algorithm learns by the feedback loop and tries to maximize the successes.

4.2 Exploration vs. Exploitation : Beta Distribution

The underlying problem is to know when to explore and exploit while selecting the gateway. Exploring means that we try a new gateway in hopes that the probability of rewards is higher even if its current reward probability is not the best. Exploitation means that we continue selecting the best performing gateway. Exploitation is the right thing to do to maximize the expected reward in the given transaction, but exploration may produce the greater total reward in the long run. If a gateway is not producing successful transactions very often and we haven't sampled it too much then probability of success is low but our confidence in the estimate is low too, so let's give it a small chance in future. However, if a gateway has processed too many transactions already and doing good, our confidence in the estimate is also high, so let's give a higher chance of being recommended.

To handle the problem of exploration and exploitation we used Beta Distribution. We used Beta Distribution for following convenient reasons.

- (1) Beta distribution can have values between 0 and 1, which is nice for our use case to represent probability of success and failure of the gateway. In general, it can present the transactional health of the gateway.
- (2) When the prior is a Beta distribution and the likelihood is a Bernoulli distribution, the posterior is also a Beta distribution.
- (3) The update formula for determining the posterior only involves addition and subtraction, whereas for other distributions it can be more complicated.

The Beta distribution has 2 parameters, α and β , which govern its shape. We refer to a specific distribution as $Beta(\alpha, \beta)$.

Initially, we set $\alpha = 1$ and $\beta = 1$. This results in a uniform distribution, i.e. we assume nothing about our chances of winning. All probabilities of winning are equally likely. This is called a minimally informative prior.

A fat distribution means more "exploration". A sharp distribution means more "exploitation" (if it has a relative high transaction health). Note that as the programmer, you don't choose whether or not to explore or exploit. You sample from the distributions, meaning it is randomly decided whether you should explore or exploit. Essentially the decision is random, with a bias toward gateways who have proven themselves to be more healthy.

4.3 Thompson Sampling

For simplicity of discussion, we first provide the details of Thompson Sampling algorithm for the Bernoulli problem, i.e. when the rewards are either 0 or 1, and for gateway i the probability of success (reward = 1) is μ_i . Next, we propose a simple new extension of this algorithm to general reward distributions with support $[0, 1]$.

The algorithm for Bernoulli gateways maintains Bayesian priors on the Bernoulli means μ_i 's. Beta distribution turns out to be a very

Figure 3: Beta Distribution with increasing β . Notice that the graph is becoming sharper towards low value of health.

Figure 4: Beta Distribution with increasing α . Notice that the graph is becoming sharper towards high value of health.

convenient choice of priors for Bernoulli rewards. Let us briefly recall that beta distributions form a family of continuous probability distributions on the interval (0, 1).

The standard *Beta* distribution gives the probability density of a value x on the interval (0,1):

$$Beta(\alpha, \beta) : prob(x|\alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)} \quad (4)$$

where B is the beta function

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt$$

. The mean of $Beta(\alpha, \beta)$ is $\alpha/(\alpha + \beta)$; and as is apparent from the pdf, higher the α, β , tighter is the concentration of $Beta(\alpha, \beta)$ around the mean. Beta distribution is useful for Bernoulli rewards because if the prior is a $Beta(\alpha, \beta)$ distribution, then after observing a Bernoulli trial, the posterior distribution is simply $Beta(\alpha+1, \beta)$ or $Beta(\alpha, \beta+1)$, depending on whether the trial resulted in a success or failure, respectively. The Thompson Sampling algorithm initially assumes gateway i to have prior $Beta(1, 1)$ on μ_i , which is natural because $Beta(1, 1)$ is the uniform distribution on (0, 1). At time t , having observed α_t successes (reward = 1) and β_t failures (reward = 0) in $k_i(t) = \alpha_t + \beta_t$ transactions of gateway i , the algorithm updates

Figure 5: Beta Distribution with increasing α . Notice that the graph is becoming sharper towards high value of health.

Table 1: Thompson Algorithm

Algorithm for Thompson Sampling
1: $\alpha_i = 0, \beta_i = 0$.
2: for $t = 1, 2, \dots$, do
3: For each gateway $i = 1, \dots, N$, sample $\theta_i(t)$ from the beta distribution with parameters $(\alpha_i + 1, \beta_i + 1)$.
4: Recommend for transaction $i(t) := \arg \max_i \theta_i(t)$ and observe reward r .
5: if $r = 1$, then $\alpha_i = \alpha_i + 1$, else $\beta_i = \beta_i + 1$.
6: end for

the distribution on μ_i as $\text{Beta}(\alpha_t + 1, \beta_t + 1)$. The algorithm then samples from these posterior distributions of the μ_i 's, and processes a transaction through a gateway according to the probability of its mean being the largest. We summarize the Thompson Sampling algorithm below (Table 1).

4.4 Non-stationary Rewards

The updating α_t and β_t can be appropriate for a stationary environment, where the probability distribution of rewards are stationary but the success rates of the gateways do not remain stationary over time, the health of a gateway keeps changing and hence the distribution is effectively non-stationary. In this case it makes sense to weight recent rewards more heavily than long-past ones. The way we have used for this to use a constant step-size parameter, and we call it as a history decay factor. The α_t and β_t are hence updated in the following way:

$$\alpha_{t+1} = \lambda(\alpha_t + r_t) \quad (5)$$

$$\beta_{t+1} = \lambda(\beta_t + r_t) \quad (6)$$

Please note that λ is the decay factor and r_t is the reward at epoch t . The decaying the history also helps in setting up an upper bound to the values of α_t and β_t . The upper bounds of the these values

Table 2: Thompson Algorithm

Algorithm for Thompson Sampling with Decay
1: $\alpha_i = 0, \beta_i = 0$.
2: for $t = 1, 2, \dots$, do
3: For each gateway $i = 1, \dots, N$, sample $\theta_i(t)$ from the beta distribution with parameters $(\alpha_i + 1, \beta_i + 1)$.
4: Recommend for transaction $i(t) := \arg \max_i \theta_i(t)$ and observe reward $\{r_{\alpha_t}, r_{\beta_t}\}$.
5: $\alpha_i = \lambda(\alpha_i + r_{\alpha_t})$, and $\beta_i = \lambda(\beta_i + r_{\beta_t})$.
6: end for

ensure the possibility of exploration. We considered the rewards to be only 0 and 1 for simplicity of the discussion but these rewards can be any value between 0 and 1 to represent various rewards. For example, in our case, the transaction failures due to risk rules don't suggest that health degradation of the gateway so its reward could vary between 0 and 1. The Thompson Algorithm with the updates can be summarized as below (Table 2):

5 RESULTS AND DISCUSSION

We have compared the success rates of the recommended and existing dynamic switching algorithms in Figure 6. The current implementation looks at the monthly, weekly and hourly success rates for each gateway for making a recommendation for the transaction. The probability of making a successful transaction is calculated using Bays' Theorem. The implementation takes the notion of discrete success rates for each gateway. The exploration for the new gateways or the gateways which were marked as down, is done on a regular interval.

- The graphs (Figure 6 and Figure 7) represent the comparison of success rates for the overall transaction traffic and for top 20 merchants at PayU.
- The data is on the basis of gateways allowed for specific merchants.
- CURRENT_SR: Success rate, as realized by current implementation. This forces the eligible payment gateway for each transaction according to the current settings.
- RECOMMENDED_SR: Success rate by the new learning algorithm.
- BEST_SR: The best possible success rate if all the eligible gateways are enabled for the merchant.
- Eligible Gateways: Every transaction has a set of eligible gateways which can process the transaction. This set depends on the payment method used by the customer and the gateways enabled on the merchant.

5.1 Impact

- The proposed implementation results into the overall success rate improvement by 3.23% from the current algorithm. This improvement is obtained without changing any arrangement for eligibility of a gateway for a transaction depending on its merchant and bin.
- The proposed implementation results into the overall success rate improvement by 8.21% if all the routable gateways are

Figure 6: Recommended vs Current Success Rates.

Figure 7: Recommended vs Current vs Best Success Rates.

enabled for all the transactions. This considers only eligible gateways for a given transaction in terms of bin processing ability of a transaction.

- The proposed implementation provides an ability to predict the success rate and the cost if a particular payment gateway

is enabled for a merchant. This can be used to give merchant visibility about enabling a particular gateway. This will also help in enabling more gateways on a merchant and take benefit of having more options while routing a transaction.

- The proposed solutions enables us to predict volume shift between various gateways in various arrangements.
- The exploration is done by the transaction with smaller amounts which reduces the loss occurred due to exploration process.
- The proposed implementation is more sensitive to changes in the health of gateways, hence it behaves better in the situations when a gateway goes down or becomes healthy.
- Cost based routing resulted into improvement of overall margins without impacting the success rates.

6 CONCLUSION

The application of reinforcement learning to the transaction routing problem has resulted into significant improvement in both success rates and margins. The reward tweaking has also contributed in further improvement of our understanding of impact of various failure reasons on overall success rates of various gateways. The growing interest for machine learning and reinforcement learning, in particular, arises from the large number of industrially relevant problems that can be modeled as a multi-path routing problems. A large number of problems in science and engineering, from robotics to game playing, tutoring systems, resource management, financial portfolio management, medical treatment design and beyond, can be characterized as sequential decision-making under uncertainty. Many interesting sequential decision-making tasks can be formulated as reinforcement learning problems. The techniques explained in this paper can be generalized to many such problems by simply modifying the model used for reward distribution.

REFERENCES

- [1] Omar Besbes, Yonatan Gur, Assaf Zeevi. *Optimal Exploration-Exploitation in a Multi-Armed-Bandit Problem with Non-stationary Rewards*.
- [2] P. Auer, N. Cesa-Bianchi, and P. Fischer. *Finite-time analysis of the multiarmed bandit problem*. Machine Learning, 47(2-3):235–256, 2002.
- [3] M. Zelen. *Play the winner rule and the controlled clinical trials*. Journal of the American Statistical Association, 64:131–146, 1969.
- [4] D. A. Berry and B. Fristedt. *Bandit problems: sequential allocation of experiments*. Chapman and Hall, 1985.
- [5] O. Chapelle and L. Li. *An empirical evaluation of thompson sampling*. In NIPS, 2011.
- [6] P. Auer, N. Cesa-Bianchi, and P. Fischer. *Finite-time analysis of the multiarmed bandit problem*. Machine Learning, 2002.
- [7] R. D. Kleinberg and T. Leighton. *The value of knowing a demand curve: Bounds on regret for online posted-price auctions*. In Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pages 594–605, 2003.
- [8] Sebastien Bubeck and Nicolo Cesa-Bianchi. *Regret analysis of stochastic and non-stochastic multi-armed bandit problems*. Foundations and Trends in Machine Learning, 5(1):1–122, 2012.
- [9] J. C. Gittins. *Bandit processes and dynamic allocation indices (with discussion)*. Journal of the Royal Statistical Society, Series B, 41:148–177, 1979.
- [10] O. Besbes, Y. Gur, and A. Zeevi. *Non-stationary stochastic optimization*. Working paper, 2014.
- [11] S. Agrawal and N. Goyal. *Analysis of thompson sampling for the multi-armed bandit problem*. CoRR, abs/1111.1797, 2011.
- [12] M. Babaioff, Y. Sharma, and A. Slivkins. *Characterizing truthful multi-armed bandit mechanisms: extended abstract*. In Tenth ACM Conference on Electronic Commerce, pages 79–88. ACM, 2009.
- [13] O. Chapelle and L. Li. *An empirical evaluation of thompson sampling*. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, Advances in Neural Information Processing Systems 24, pages 2249–2257. Curran Associates, Inc., 2011.
- [14] N. Gatti, A. Lazaric, and F. Trovati. *A truthful learning mechanism for contextual multi-slot sponsored search auctions with externalities*. In Thirteenth ACM Conference on Electronic Commerce, pages 605–622, 2012.
- [15] O.-C. Granmo. *Solving two-armed bernoulli bandit problems using a bayesian learning automaton*. International Journal of Intelligent Computing and Cybernetics, 3(2):207–234, 2010.
- [16] S. Scott. *A modern bayesian look at the multi-armed bandit*. Applied Stochastic Models in Business and Industry, 26:639–658, 2010.
- [17] D. Bergemann and J. Valimaki. *Learning and strategic pricing*. Econometrica, 64:1125–1149, 1996.
- [18] D. Bergemann and U. Hege. *The financing of innovation: Learning and stopping*. RAND Journal of Economics, 36 (4):719–752, 2005.
- [19] B. Awerbuch and R. D. Kleinberg. *Addaptive routing with end-to-end feedback: distributed learning and geometric approaches*. In Proceedings of the 36th ACM Symposium on Theory of Computing (STOC), pages 45–53, 2004.
- [20] F. Caro and G. Gallien. *Dynamic assortment with demand learning for seasonal consumer goods*. Management Science, 53:276–292, 2007.
- [21] S. Pandey, D. Agarwal, D. Chakrabarti, and V. Josifovski. *Bandits for taxonomies: A model-based approach*. In SIAM International Conference on Data Mining, 2007.