Bayesian Optimal Policies for Asynchronous Bandits with Known Trends

Mohammed-Amine Alaoui MD.AMINE.ALAOUI@GMAIL.COM
Tanguy Urvoy TANGUY.URVOY@ORANGE.COM
Fabrice Clérot FABRICE.CLEROT@ORANGE.COM
Orange-Labs

Abstract

We formalize the *asynchronous multi-armed bandits with known trend* problem (AMABKT) and propose a few empirical solutions, the most efficient one being based on finite-horizon Gittins indices. We designed a GPU-optimized code which allowed us to solve exhaustively the – intractable – Bayesian Optimal policy for up to four arms with reasonable horizons. Our experiments underline the quality of the trend-aware finite-horizon Gittins index as an approximant for the Bayesian optimal policy on stochastic AMABKTs.

Keywords: reinforcement learning, value-iteration, multi-armed bandits, scheduling, GPU

1. Introduction

Consider the following *user-interface selection problem*: an engineer has at her disposal several versions of a user interface for a device or a web site, she wants to deploy the most efficient one on the long run. The old-fashioned approach would be to first test these interfaces on a panel of users then deploy the one that performed the best. However, it is a long, heavy, and costly process: a better option is certainly to favor a '*continuous improvement process*' through a well adapted sequential experiment design. The *Multi-Armed Bandit* problem (MAB) is a natural candidate to model this problem. This old *exploration/exploitation* model was first introduced to optimize medical experiments (Thompson, 1933; Lai and Robbins, 1985), but MAB algorithms turned out to contribute as building blocks to several modern applications and have since been deeply studied (see Cesa-Bianchi and Lugosi, 2006; Bubeck and Cesa-Bianchi, 2012, for extended surveys). The user-interface selection problem presents however two specificities:

1. When a new interface is proposed to the users, they need some time to adapt: the rewards processes are **not stationary** and they follow a *"learning curve"*.
2. When users practice a given interface, they do not progress on the other ones: the reward processes evolve in an **asynchronous** manner.

The user-interface selection problem is not the only industrial application with these properties. We can mention for instance the recommendation of items to users who may get bored of a topic (Hu and Ogihara, 2011), the selection of distributed sensors with decreasing batteries (Tran-Thanh et al., 2010), or the crowdsourcing of workers who first improve at a new task only to get bored ultimately (Tran-Thanh et al., 2014).

As we will see in section 1.2, several algorithms have been proposed recently to handle non-stationary but synchronous reward processes. In order to handle the asynchronicity we propose to go back to plain old *Markovian bandits* and fish in a more classical, and more Bayesian, literature (Berry and Fristedt, 1985; Gittins, 1989; Mahajan and Teneketzis, 2008). In this literature, a MAB is a scheduling problem between several – known – independent *Markov Processes*. At each stage, the learner is asked to decide which process will evolve while the others stay frozen. This asynchronous behavior leads to a clear distinction between the individual *process times* (i.e the number of time an arm has been pulled), and the *global time*. This asyncronous product of Markovian processes is often refered as *rested bandits* by opposition to the general case of *restless bandits* where the other arms may (or may not) evolve when we pull an arm.

As we will see in section 2, there is a risk of confusion in the literature between the arms when refered as – unknown – reward processes like in (Lai and Robbins, 1985), and the arms when refered as – know – Bayesian knowledge aquisition processes like in (Gittins, 1989)[1]. To the best of our knowledge, the notion of bandits with asynchronous reward processes is new.

1.1. Problem Statement

1.1.1. General Formalization of the Problem

We will call *asynchronous multi-armed bandit* (AMAB) a MAB where the reward of each arm $a = 1, \ldots, K$ is modeled by a sequence of bounded rewards $x_a(1), \ldots, x_a(T)$. A policy is defined by a sequence of arms $A(t) \in \{1, \ldots, K\}$ and the reward of this policy till time t is $\sum_{a=1}^{K} \sum_{n=1}^{n_a(t)} x_a(n)$ where $n_a(t) = \sum_{s=1}^{t} [\![A(s) = a]\!]$ is the *process time* of arm a (We use $[\![\cdot]\!]$ for the indicator function). Note that a policy is entirely defined by its corresponding process time vectors $\mathbf{n}(t) = (n_1(t), \ldots, n_K(t))$, with *global time* $t = \sum_a n_a(t)$. The performance of a given policy $\mathbf{n}$ is evaluated through its *regret* which measures the opportunity loss at time t against an informed reference policy $\mathbf{n}^*$:

$$\mathrm{Regret}_{\mathbf{n}}(t) = \sum_{a=1}^{K} \left[\sum_{s=1}^{n_a^*(t)} x_a(s) - \sum_{s=1}^{n_a(t)} x_a(s) \right] .$$

1.1.2. Stochastic Asynchronous Bandits with Known Trends

To simplify a bit, we will assume the rewards sequences to be defined by independent Bernoulli processes which are modulated by – known – trends functions. More formally we will assume that: $x_a(n) = f_a(n) \cdot Y_a$ where $Y_a \sim \mathrm{Bern}(\mu_a)$[2].

The assumption of known trends may seem strong at first sight but it is important to note that stochastic MABs are special cases or our setting with the same known – constant – trend for all the arms. A specificity of Asynchronous MABs with Known Trends (AMABKT) is that the optimal strategy is not, as in classical MABs, to pull indefinitely the same arm.

1. With that distinction in mind, the reward processes of synchronous MABs should be considered as "purely restless" even if their corresponding Bayesian knowledge-aquisition processes are "rested".
2. Another modeling option could have been to modulate the Bernoulli parameters: $x_a(n) = X_a$ where $X_a \sim \mathrm{Bern}(f_a(n) \cdot \mu_a)$, but we kept the first option for the sake of simplicity.

Indeed, with $F_a(n) = \sum_{s=1}^{n} f_a(s)$ denoting the *cumulative trend*, the optimal policy in expectation at time t is to play:

$$\mathbf{n}^*(t) = \operatorname*{arg\,max}_{n_1+\ldots+n_K=t} \sum_a F_a(n_a) \cdot \mu_a \ . \tag{1}$$

The expected regret of a policy $\mathbf{n}$ hence simplifies into:

$$\mathbb{E}\left[\text{Regret}_{\mathbf{n}}(t)\right] = \max_{m_1+\ldots+m_K=t} \sum_a \left[F_a(m_a) - F_a(n_a)\right] \cdot \mu_a \ . \tag{2}$$

Obviously, if the trends have bounded supports, all policies are asymptotically equivalent. Moreover, there may not even exist a good anytime policy: take for instance a two-armed AMABKT with parameters $\mu_1 > \mu_2 > 0$. If the trend f is identical for all arms and non-decreasing, the optimal and anytime policy will be to stick on the first arm, but if instead we have the "gear-shaped" trend $f(n) = [\![n < T \vee n > 2T]\!]$, then the optimal policy at time $2T$ is (T, T) while the optimal policy at time $3T$ is $(3T, 0)$. If a policy is optimal at time $2T$, it cannot be optimal at time $3T$: there is hence no anytime optimal policy.

1.2. Related Works

As mentioned in the Introduction, several formalizations and policies have been proposed to tackle synchronized non-stationary rewards (see for instance Kocsis and Szepesvári, 2006; Garivier and Moulines, 2011; Besbes et al., 2014; Allesiardo et al., 2016). The most extreme case of a non-stationarity being the adversarial MAB of Auer et al. (2002b). The synchronicity is a key property for the $\Theta(\sqrt{KT})$ regret bound (against the best single arm in hindsight) for oblivious adversarial MABs. Consider for instance a two-armed AMAB defined by $x_1(n) = [\![n \geq T/2]\!]$ and $x_2(n) = x_1(n)/2$. Before getting any reward/feedback from one arm, one must pull it at least $T/2$ times. Reaching this point, with horizon set to T, it will be too late to switch to the other arm. The optimal policy is hence to pick one arm blindly and play it till the end: the expected regret for an adversarial AMABKT is in $\Omega(T)$.

We focus here on models where the arms rewards are asynchronous. The *budgeted bandits* model was introduced by Tran-Thanh et al. (2010) to model sensor networks with limited battery. In their formalization, the budget is shared between the arms like in a knapsack problem. The *mortal bandits* model introduced by Chakrabarti et al. (2009) assumes that each arm has a known budget of pulls (or a known probability of expiring). This assumption of a finite-budget for each arm is a special case of a known decreasing "step-shaped" trend. Their analysis is focused on reward.

In *scratch games*, a similar model introduced by Féraud and Urvoy (2013), each arm also has its own budget. The reference policy used in their analysis is the one which first pull the best arm, then proceeds to the second best, and so on till the end of all the budgets. In (Bouneffouf and Féraud, 2016), they consider an AMABKT where all arms are modulated by the same smooth (Lipschitzian) trend. They propose an adaptation of the UCB algorithm called A-UCB where the player picks the arm maximizing the index $[\hat{\mu}_a(n_a) + \sqrt{2\log t/n_a}] \cdot f(n_a(t))$. They obtain an anytime logarithmic regret upper bound. The anytime policy they use as a informed reference is however far from being a match against the optimal policy as defined in (1).

2. Bayesian Optimal Policy

The proper way to approach stochastic MABs with MDPs is to encode the knowledge we have on each arm into the states of a known Markov chain.

For a stochastic MAB with Bernoulli reward laws, we will assume the reward parameters μ_i to be drawn from a Beta prior distribution. With Bayesian updating, the posterior law of an arm a for which we encountered α_a wins and β_a losses[3] will be the distribution $\text{Beta}(\alpha_a, \beta_a)$. We will hence code each state $s \in \mathcal{S}$ of the MDP as a tuple $(\alpha_1, \beta_1, \ldots, \alpha_K, \beta_K)$ with a winning probability $\alpha_a/(\alpha_a + \beta_a)$ for playing arm a. A winning transition $s \xrightarrow{a} s'$ will increment α_a and give a reward $r_a(s,s') = +1$ while a loosing one will increment β_a and gives no reward. The (Beta) Bayesian optimal policy till horizon T for the Bernoulli MAB is obtained by solving this MDP through undiscounted *value-iteration.* Using standard notations (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998), the optimal value and policy for a state s at time t are given by:

$$V^*(s,t) = \begin{cases} \max\limits_a \sum_{s' \in \mathcal{S}} p_a(s,s')\,(r_a(s,s') + V^*(s', t+1)) & \text{if } t < T, \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

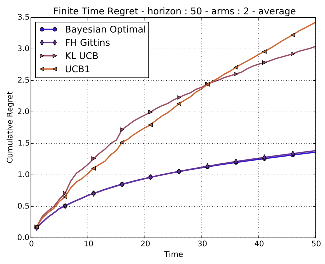
$$\pi^*(s,t) = \arg\max_a \sum_{s' \in \mathcal{S}} p_a(s,s')\,\big(r_a(s,s') + V^*(s', t+1)\big) \ \text{ if } t < T \tag{4}$$

The number of states of this MDP is in $\Omega(T^{2K})$ which makes the exhaustive computation of the optimal Bayesian policy intractable. We however worked on a highly optimized parallel `CUDA+openMP` code to compute the optimal policy on a compressed MDP. The technical aspects of this code are detailed in Appendices A and B. We were able to compute exhaustively a reference policy for up to four arms and reasonable horizons. Our experiments summarized on Figure 1 confirms experimentally that the Bayesian optimal policy steadily outperforms the state-of-the-art stochastic bandits algorithms, namely UCB1 (Auer et al., 2002a) and KL-UCB (Garivier and Cappé, 2011). Similar experiments were performed in (Ginebra and Clayton, 1999).

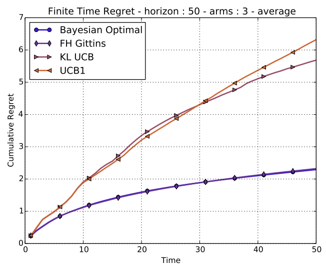
2.1. The Gittin's index theorem

The most remarkable result on Markov Bandits is the possibility that we have to simplify its optimal MDP policy into an explicit and tractable *index-maximization* policy (Bradt et al., 1956; Gittins, 1989). This index can be interpreted as the price a rational player would be willing to pay in order to play a given arm. This powerful indexability result is only valid for infinite horizon with discounted rewards, but a non-discounted finite time index (FH-Gittins) has also been considered as good approximation for Bernoulli bandits (Niño-Mora, 2011; Kaufmann, 2016), or for Gaussian bandits for which a regret analysis has been recently provided by Lattimore (2016). We empirically confirm the excellent quality of this approximation for Bernoulli MABs on Figure 1. Especially on Subfigure 1(*e*) where its best-arm error rate (blue chart) is shown to evolve from 0.05% to virtually 0% of the states when it approaches the horizon.

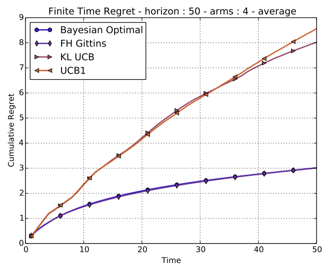
3. These are in fact pseudo-counts if we take the $\text{Beta}(1,1)$ prior into account.



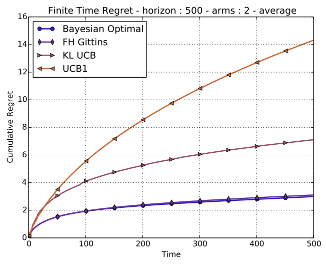
(a) 2 arms - horizon 50



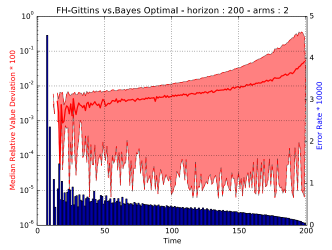
(b) 3 arms - horizon 50



(c) 4 arms - horizon 50



(d) 2 arms - horizon 500



(e) FH-Gittins error

Figure 1: Comparison of different algorithms against Bayesian optimal policy. The regret as defined in (2) is averaged over 100×1000 uniform MAB instances. With 4 arms and horizon 50 the non-compressed MDP counts $2 \cdot 10^{12}$ states. Subfigure 1(e) measures the approximation error of FH-Gittins v.s. optimal policy for 2 arms and horizon 200.

3. Trend-aware Bayesian Policies

Adapting the Bayesian optimal policy to the Bernoulli AMABKT problem is straightforward, one only needs to modulate the rewards of equations (3) and (4) by the trend functions:

$$r_a((\ldots,\alpha_a,\beta_a,\ldots),(\ldots,\alpha_a+1,\beta_a,\ldots)) := f_a(1+\sum_i(\alpha_i+\beta_i)) \qquad \textit{winning transition}$$

$$r_a((\ldots,\alpha_a,\beta_a,\ldots),(\ldots,\alpha_a,\beta_a+1,\ldots)) := 0$$

A more scalable "trend-aware" FH-Gittins index can also be computed by solving the following calibration problem encoded as a "risky single-arm" versus "constant reward" MDP (n is the number of rounds left till the horizon, τ is the stopping time):

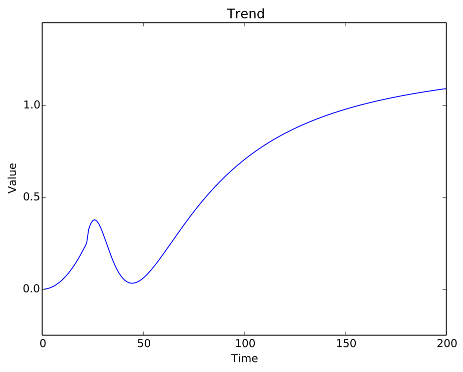
$$\mathrm{FHG}_a(\alpha,\beta,n) = \sup\left\{\lambda\in[0,1] \,:\, n\lambda \le \sup_{0<\tau<n} \mathbb{E}_{\alpha,\beta}\left[\sum_{s=0}^{\tau-1} Y_a(s)\cdot f_a(s) + (n-\tau)\lambda\right]\right\} \tag{5}$$

4. Experiment

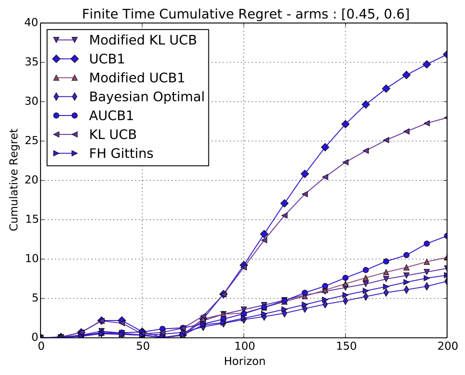
The trend-aware FH-Gittins index being an optimistic estimate of the future arm values, it makes sense to use a similar approach to adapt existing anytime indices like UCB1:

$$\text{modified-UCB}(a,t,n) = \text{UCB}(a,t)\cdot[F_a(n_a(t)) - F_a(n+n_a(t))] \tag{6}$$

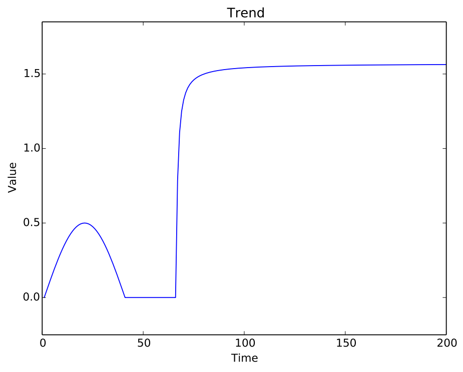
We applied the same recipe to kl-ucb. Our experiments are summarized in Figure 2. As expected, the two non-adapted anytime policies fails to exploit the trend knowledge. The a-ucb algorithm does reasonably well as soon as the trend is kept smooth enough to keep the local trend value correlated to its future, but modified ucb and kl-ucb fares better. The adapted fh-Gittins index is shown to give a really tight approximation of the Bayesian Optimal policy.



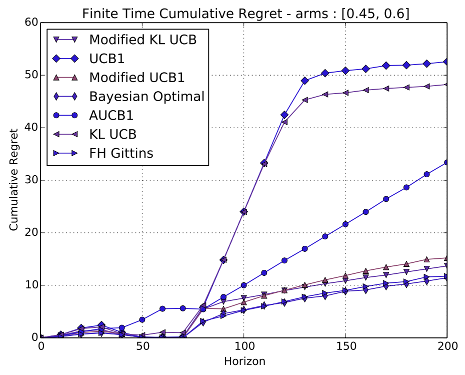
(a)



(b)



(c)



(d)

Figure 2: Experiment: the trend on the left is the same for all the arms and the regret, defined as in Equation (2), is averaged 10000 times on a 2-armed bandit problem 0.45 and 0.6).

5. Conclusion and Future Works

This preliminary empirical study on Multi-Armed Bandits with Known Trends is promising. We managed to compute the Bayesian optimal policy sufficiently far to get a solid reference to evaluate approximate index policies. Even with our GPU-optimized code, this optimal policy remains intractable in practice, but the adapted fh-Gittins index we propose is a good candidate for applications. As a future work we ambition to provide a regret analysis for the proposed solutions. Another research avenue we considered is the use of approximate MDP algorithms for indexes estimation (see Alaoui, 2016).

References

Mohammed-Amine Alaoui. Reinforcement learning and optimal strategies on the gpu. Technical report, Orange-Labs, Centrale-Supéléc, 2016.

Robin Allesiardo, Raphaël Féraud, and Odalric-Ambrym Maillard. Random shuffling and resets for the non-stationary stochastic bandit problem. 2016. URL http://128.84.21.199/abs/1609.02139.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002a. ISSN 0885-6125. doi: 10.1023/A:1013689704352.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002b.

D. A. Berry and B. Fristedt. *Bandit Problems: Sequential Allocation of Experiments.* Chapman and Hall, London, 1985.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1st edition, 1996. ISBN 1886529108.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in Neural Information Processing Systems 27*, pages 199–207. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5378-stochastic-multi-armed-bandit-problem-with-non-stationary-rewards.pdf.

Djallel Bouneffouf and Raphael Féraud. Multi-armed bandit problem with known trend. *Neurocomputing*, 205:16–21, 2016. ISSN 0925-2312. doi: 10.1016/j.neucom.2016.02.052. URL http://www.sciencedirect.com/science/article/pii/S092523121600299X.

Russell N Bradt, SM Johnson, and Samuel Karlin. On sequential designs for maximizing the sum of n observations. *The Annals of Mathematical Statistics*, 27(4):1060–1074, 1956.

Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012. doi: 10.1561/2200000024.

Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, New York, NY, USA, 2006. ISBN 0521841089.

Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 273–280, 2009.

Raphaël Féraud and Tanguy Urvoy. Exploration and exploitation of scratch games. *Machine Learning*, 92(2):377–401, 2013. ISSN 1573-0565. doi: 10.1007/s10994-013-5359-2.

Aurélien Garivier and Olivier Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. In *COLT*, volume 19 of *JMLR Proceedings*, pages 359–376. JMLR.org, 2011. URL http://dblp.uni-trier.de/db/journals/jmlr/jmlrp19.html#GarivierC11.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems. In *Algorithmic Learning Theory*, pages 174–188, 2011.

Josep Ginebra and Murray K Clayton. Small-sample performance of bernoulli two-armed bandit bayesian strategies. *Journal of statistical planning and inference*, 79(1):107–122, 1999.

J.C. Gittins. *Multi-armed Bandit Allocation Indices*. Wiley-Interscience series in systems and optimization. Wiley, Chichester, NY, 1989.

Yajie Hu and Mitsunori Ogihara. *Nextone player: A music recommendation system based on user behavior*, pages 103–108. 2011. ISBN 9780615548654.

Emilie Kaufmann. On Bayesian index policies for sequential resource allocation. working paper or preprint, January 2016. URL https://hal.archives-ouvertes.fr/hal-01251606.

Levente Kocsis and Csaba Szepesvári. Discounted ucb. In *2nd PASCAL Challenges Workshop*, 2006.

T.L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

Tor Lattimore. Regret analysis of the finite-horizon gittins index strategy for multi-armed bandits. In *Proceedings of the 29^{th} Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 1214–1245, 2016. URL http://jmlr.org/proceedings/papers/v49/lattimore16.html.

Yuxi Li and Dale Schuurmans. Mapreduce for parallel reinforcement learning. In *Proceedings of the 9th European Conference on Recent Advances in Reinforcement Learning*, EWRL'11, pages 309–320, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-29945-2. doi: 10.1007/978-3-642-29946-9_30. URL http://dx.doi.org/10.1007/978-3-642-29946-9_30.

Aditya Mahajan and Demosthenis Teneketzis. *Multi-Armed Bandit Problems*, pages 121–151. Springer US, Boston, MA, 2008. ISBN 978-0-387-49819-5. doi: 10.1007/978-0-387-49819-5_6.

José Niño-Mora. Computing a classic index for finite-horizon bandits. *INFORMS Journal on Computing*, 23(2):254–267, 2011.

Jorn H Postma. Speeding up reinforcement learning with graphics processing units. Master's thesis, TU Delft, Delft University of Technology, 2015.

Ronald L. Rivest and Yiqun Yin. Simulation results for a new two-armed bandit heuristic. In *Proceedings of a Workshop on Computational Learning Theory and Natural Learning Systems (Vol. 1) : Constraints and Prospects: Constraints and Prospects*, pages 477–486, Cambridge, MA, USA, 1994. MIT Press. ISBN 0-262-58126-4. URL http://dl.acm.org/citation.cfm?id=192827.188551.

R.S. Sutton and A.G. Barto. *Reinforcement Learning*. The MIT Press, 1998.

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

Long Tran-Thanh, Archie Chapman, Enrique Munoz de Cote, Alex Rogers, and Nicholas R. Jennings. Epsilon-first policies for budget limited multi-armed bandits. In *AAAI*, April 2010. URL http://eprints.ecs.soton.ac.uk/20806/6/AAAI2010_Tran-Thanh.pdf.

Long Tran-Thanh, Sebastian Stein, Alex Rogers, and Nicholas R Jennings. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *Artificial Intelligence*, 214:89–111, 2014.

Appendix A. Efficient Implementation on GPU

The MDP defined in Section 2 being a directed acyclic graph, value iteration proceeds in an incremental fashion starting from the horizon toward the initial state. The optimal values of states in stage $t+1$ are used to evaluate those of states in stage t. This local dependency in addition to the possibility of independently generating each stage, open the doors to pipe-lining generation, the stage-wise value iteration and an eventual serialization process. In practice, the stages can be compressed by normalizing lexicographically the arms (α_a, β_a) (See Fig. 4). Value iteration being embarrassingly parallel, the computation power of GPUs can be utilized with ease, provided that the implementation of data structures is well suited to the physical constraints of the GPU architecture, namely memory coalescing and minimal thread divergence. The sequential generation (Alg. 1) and serialization processes can be performed by the more adapted CPU architecture. On another note, a thorough study of the structure of the stages in the bandit's MDP showed interesting relations between successive stages, which in turn render the generation process embarrassingly parallel as well, ushering in a more scalable Bayesian optimal policy computation. Figure 5 illustrates this relations for 2 and 3 arms. More generally, for a k-armed problem, stage t can be generated parallely from stage $t+k$ in a similar fashion. Appendix B addresses in greater details the relations between stages for a 2 armed problem.

Appendix B. Stages Encoding for a 2-armed Bandit Problem

This section studies the structure of stages in 2-armed problems. It aims at inducing relations between successive ordered stages so as to parallely generate if the other is already generated. To do so, an encoding of 2-armed stages is introduced, this latter shows interesting and generalizable properties regarding the structure and links between successive states.

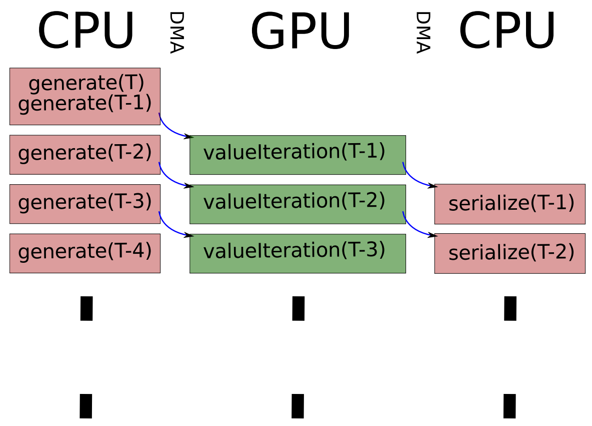
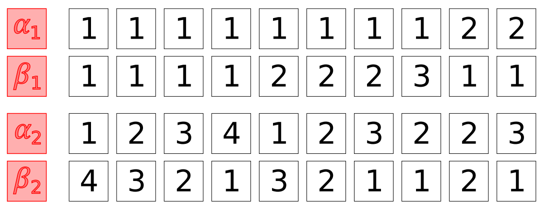
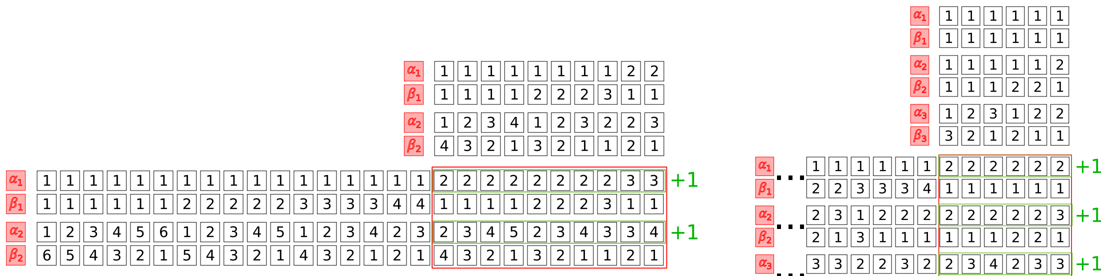


Figure 3: Pipelining value iteration.



α_1	1	1	1	1	1	1	1	1	2	2
β_1	1	1	1	1	2	2	2	3	1	1
α_2	1	2	3	4	1	2	3	2	2	3
β_2	4	3	2	1	3	2	1	1	2	1

Figure 4: A two-armed bandit stage as a sequence of lexically ordered and normalized states. In term of value iteration the two states $\{1,1,1,4\}$ and $\{1,4,1,1\}$ are equivalent. Note that on stage t for each state $\{\alpha_1,\beta_1,\alpha_2,\beta_2\}$ we must have $\sum_i \alpha_i+\beta_i = t+4$ (+4 is for the prior). Here $t=4$.



(a) stage 4 and 6 for 2 arms

(b) stage 3 and stage 6 for 3 arms

Figure 5: Tiled link between two successive stages.

Algorithm 1: Sequential Generation

```
Function generateEqual($i, j, arms, t, state, stage$)
    if $arms = 1$ then
        if $t - i \geq j$ then
            $state$.Append(($i, t - i$));
            $stage$.Append($state$);
        end
        else
            for $l = j, \dots, t$ do
                $copy = state$.Copy();
                $copy$.Append(($i, l$));
                generateEqual($i, l, arms - 1, t - i - l, copy, stage$);
                generateIncremental($i + 1, 1, arms - 1, t - i - l, copy, stage$);
            end
        end
    end
Function generateIncremental($i, j, arms, t, state, stage$)
    if $arms = 1$ then
        for $m = i, \dots, t$ do
            $copy = state$.Copy();
            $copy$.Append(($m, t - m$));
            $stage$.Append($copy$);
        end
    end
    else
        for $k = i, \dots, t$ do
            for $l = j, \dots, t$ do
                $copy = state$.Copy();
                $copy$.Append(($i, l$));
                generateEqual($k, l, arms - 1, t - k - l, copy, stage$);
                generateIncremental($k + 1, 1, arms - 1, t - k - l, copy, stage$);
            end
        end
    end
Function generate($K, t$)
    $stage \leftarrow$ empty stage , $state \leftarrow$ empty state;
    generateIncremental($1, 1, K, t, state, stage$);
```

A 2-armed problem stage is comprised of a sequence of normalized states as illustrated by Fig. 4. In the following, $\{\alpha_1, \beta_1, \alpha_2, \beta_2\}$ denotes a state in such a stage.

Property 1 (Number of different unordered states) *For a 2-armed bandit problem, the number of different unordered states a stage t is given by:*

$$N^*(t) = \begin{cases} \frac{1}{2}\binom{2K-1+t}{2K-1} & \text{if } 2K+t \text{ is even,} \\ \frac{1}{2}\left[\binom{2K-1+t}{2K-1} + \frac{2K+t}{2} - 1\right] & \text{otherwise} \end{cases}$$

and more generally, the number of different unordered states for a K-armed problem is:

$$N^*(t) \simeq \frac{N(t)}{K}$$

Definition 1 (Encoding) *An encoding of a stage t is a matrix $\mathcal{M}(t)$ where the $\mathcal{M}(t)_{i,j}$ is the number of states where $\alpha_1 = i$ and $\beta_1 = j$* [4].

Remark 2 *For this code to completely characterize the associated stage it must be able to determine the possible values of α_2 (β_2 is then deduced from the values of the other parameters and the time step t). The next property describes these values and hence the decoding process.*

Example 1 *The encoding of the stage of Fig. 4 is given by:*

$$\begin{pmatrix} 4 & 3 & 1 \\ 2 & 0 & 0 \end{pmatrix}$$

Property 2 (Decoding) *For each α_1 and β_1, α_2 take its values in a sequence of $\mathcal{M}(t)_{i,j}$ numbers. Furthermore, these numbers are consecutive and the first one, $\alpha_2^{initial}$, depends on $\mathcal{M}(t)_{i,j-1}$ as follows:*

$$\alpha_2^{initial} = \begin{cases} i & \text{if } \mathcal{M}(t)_{i,j-1} = \mathcal{M}(t)_{i,j} + 1 \text{ or } j = 1, \\ i+1 & \text{otherwise} \end{cases}$$

Example 2 *In the stage of Fig. 4, α_2 starts from the value of α_1 except for states $\alpha_1 = 1, \beta_2 = 3$ (only one state) for which $\alpha_2^{initial} = 1 + 1 = 2$. And indeed, the difference between $\mathcal{M}(t)_{1,2} = 3$ and $\mathcal{M}(t)_{1,3} = 1$ is greater than one.*

Property 3 (Structure of a row) $(\mathcal{M}(t)_{i,j})_j$ *is a sequence of successive and decreasing numbers except for at most one given index $J(t, i)$ for which $\mathcal{M}(t)_{i,J-1} = \mathcal{M}_{i,J} + 2$. This sequence starts from $\mathcal{M}(t)_{i,1} = t - 2(i - 1)$, and the following holds:*

$$J(t, i) = \left\lceil \frac{t - 2(i-1)}{2} \right\rceil + 1$$

4. i and j start from 1

Property 4 (Dimensions of the code matrix) *The number of rows ,n, and the number of non null entries per row ,m, depend both on t and are given by:*

$$n(t) = \left\lceil \frac{t}{2} \right\rceil$$

$$m(t,i) = \begin{cases} 1 & \textit{if } t = 1, 2 \textit{ and } i = 1, \\ t-1 & \textit{if } t > 2 \textit{ and } i = 1, \\ m(t-2, i-1) & \textit{if } t > 2 \textit{ and } i > 1 \end{cases}$$

Table 1 contains the code of stages from 1 to 7. It also reveals interesting relations between $\mathcal{M}(t)$, $\mathcal{M}(t-1)$ and $\mathcal{M}(t-2)$ which are formalized by the following property:

Property 5 (Relations between successive codes) *The following holds:*

$$\mathcal{M}(t+1)_{i,j+1} = \mathcal{M}(t)_{i,j} \textit{ for } i \leq n(t) \textit{ and } j \leq m(t,i)$$

$$\mathcal{M}(t+2)_{i+1,j} = \mathcal{M}(t)_{i,j} \textit{ for } i \leq n(t) \textit{ and } j \leq m(t,i)$$

Property 6 (Number of states such as $\alpha_1 = 1$) *The number of such states in stage t is the sum of all the coefficients of the first row of $\mathcal{M}(t)$, given by the following recursive equations:*

$$N_1(t) = \begin{cases} 1 & \textit{if } t = 1, \\ N_1(t-1) + t - 1 & \textit{if } t \textit{ is even and } t > 1, \\ N_1(t-1) + t & \textit{otherwise} \end{cases}$$

And equivalently by:

$$N_1(t) = \begin{cases} 2(\frac{t}{2} - 1)(\frac{t}{2} + 1) + 2 & \textit{if } t \textit{ is even}, \\ 2\frac{t-1}{2}(\frac{t-1}{2} + 1) + 1 & \textit{otherwise} \end{cases}$$

Remark 3 *The sum of all the coefficients of the matrix $\mathcal{M}(t)$ must equate the number of normalized states $N^*(t)$, i.e.:*

$$N^*(t) = \sum_{i=1}^{n(t)} N_i(t)$$

the relations between $\mathcal{M}(t)$ and $\mathcal{M}(t-2)$ allow us to transform the equation above to:

$$N^*(t) = \sum_{i}^{n(t)} N_1(t - 2(i-1))$$

Which can be proven using the explicit expression of $N_1(t)$ above and some computations.

	(1)
(2)	$\begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}$
$\begin{pmatrix} 4 & 3 & 1 \\ 2 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 5 & 4 & 3 & 1 \\ 3 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$
$\begin{pmatrix} 6 & 5 & 4 & 2 & 1 \\ 4 & 3 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 7 & 6 & 5 & 4 & 2 & 1 \\ 5 & 4 & 3 & 1 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Table 1: Codes for some stages