A learning algorithm for the Whittle index policy for scheduling web crawlers
Konstantin Avrachenkov, Vivek S Borkar

To cite this version:

HAL Id: hal-02416599
https://inria.hal.science/hal-02416599
Submitted on 17 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A learning algorithm for the Whittle index policy for scheduling web crawlers

Konstantin Avrachenkov
INRIA Sophia Antipolis,
2004, Route des Lucioles, B.P.93,
06902, Sophia Antipolis, France
E-mail: k.avrachenkov@inria.fr

Vivek S. Borkar
Department of Electrical Engineering,
Indian Institute of Technology Bombay,
Powai, Mumbai 400076, India.
E-mail: borkar.vs@gmail.com

Abstract—We revisit the Whittle index policy for scheduling web crawlers for ephemeral content proposed in Avrachenkov and Borkar, IEEE Trans. Control of Network Systems 5(1), 2016, and develop a reinforcement learning scheme for it based on LSPE(0). The scheme leverages the known structural properties of the Whittle index policy.

I. INTRODUCTION

Reinforcement learning for approximate dynamic programming has been a popular approach to approximate policy evaluation or computation for Markov decision processes with large state spaces. A data driven iterative scheme based on real or simulated data, it offers the possibility of working around, to a significant extent, the so called curse of dimensionality which these problems suffer from. A further and very significant reduction in complexity is possible if one exploits the explicitly known structural properties, if any, of the problem. Recently there have been a few works that do so, e.g., exploiting threshold nature of the policy [11], [16], parametrized threshold policies [19], or index policies such as the Whittle policy for restless bandits [5]. In particular, [5] addressed the problem of learning the Whittle policy when Whittle indexability is known, but structure of the index is not explicitly known. This work addresses a situation when the latter is known in a parametric form, but the parameters are unknown, and the algorithm is required to operate online with streaming real time data. As a test case, we consider the specific problem of scheduling web crawlers for ephemeral content analyzed in [2] (see also [3], [14] for related work). We describe this problem in the next section and summarize the main results of [2]. Some preliminary analysis of the problem is given in section III. This is followed by the proposed learning scheme in section IV. Section V presents some supporting numerical experiments.

II. PROBLEM FORMULATION

Consider crawling decisions at discrete times 0, 1, 2, · · ·. The content arrives in continuous time according to a Poisson process at site $i$ with rate $\lambda_i$ for $1 \leq i \leq N$. Content at site $i$ more than $M_i$ time units old is dropped. The interest in the content at site $i$ less than $M_i$ time units old decays exponentially with rate $c_i > 0$ [6], [20], [12]. Let $X_i(n)$ denote the cumulative interest in content at site $i$ at time $n$. Consider $\zeta$ items arriving at site $i$ during the time interval $[m, n)$ at (random) times $\{\tau(i), 1 \leq i \leq \zeta\}$. Then $\{\tau(i)\}$ are IID uniform on $[m, n]$ conditioned on $\zeta = \text{(say)} k$, and

$$p_i(k) := P(\zeta = k) = \frac{(\lambda_i(n - m))^k}{k!} e^{-\lambda_i(n-m)}.$$  

Define

$$\alpha_i = e^{-c_i},$$  

$$Z_i(n) := \sum_{\{m:n-1 \leq \tau_i(m) < n\}} \alpha_i^{n-\tau(m)},$$  

$$X_i(n) = \sum_{m=n-M+1}^{n} \alpha_i^{n-m} Z_i(m),$$  

$$\xi_i(n) = Z_i(n) - \alpha_i^{M_i} Z_i(n-M_i),$$  

$$u_i = u_i(\lambda_i, \alpha_i) := E[\xi_i(n)].$$
The objective is to maximize the average interest level zero, so that knowledge is no longer relevant. The state is instantly reset to zero at time $n$, where $N_x$ crawlers can be active at any time.

Thus the observed dynamics is effectively the deterministic one given by (2)-(3): when you crawl, you do observe the actual random state at the end of the interval, but the state is instantly reset to zero, so that knowledge is no longer relevant. The objective is to maximize the average interest level

$$\lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} \sum_{i=1}^{N} X_i(n) \mu_i(n)$$

where $\mu_i(n) = 1$ if location $i$ is crawled at time $n$ and 0 otherwise, subject to the constraint that only $N_0 < N$ crawlers can be active at any time. This can be cast as a restless bandit problem that is Whittle indexable [2]. The Whittle index is given by [2]

$$\gamma_i(x, \lambda_i, \alpha_i) := \frac{1}{C_i} \left[ \zeta_i(x, \lambda_i, \alpha_i)((1 - \alpha_i)x - u_i) + \left(1 - \alpha_i \frac{\zeta_i(x, \lambda_i, \alpha_i)}{1 - \alpha_i}\right) u_i \right]$$

where

$$\zeta_i(x, \lambda_i, \alpha_i) := \left[ \log_{\alpha_i} \left( \frac{u_i - (1 - \alpha_i)x}{u_i} \right) \right].$$

Here $\lceil \cdot \rceil$ stands for ‘the least integer exceeding’ and we have rendered explicit the dependence on the unknown arrival rate $\lambda_i$ and the unknown decay rate $\alpha_i$ via $u_i$. Let

$$\lambda := [\lambda_1, \ldots, \lambda_N] \in \mathcal{R}^N, \alpha := [\alpha_1, \ldots, \alpha_N] \in (0, 1)^N, \Gamma := [\lambda : \alpha] \in \mathcal{R}^N \times (0, 1)^N,$$

$$u(\Gamma) := [u_1(\lambda_1, \alpha_1), \ldots, u_N(\lambda_N, \alpha_N)]^T.$$

Anticipating the randomized policies we consider later, denote the reward under the Whittle index policy by $J(\Gamma) := \sum_{i=1}^{N} E_i [X_i(n)]$, where $E_i [\cdot]$ denotes the stationary expectation. Let $\pi_T(dx)$ denote the stationary distribution of $X(n) = [X_1(n), \ldots, X_N(n)]$, $n \geq 0$. We shall denote by $\nabla y$ the gradient w.r.t. the variable $y$.

The Whittle policy is as follows: At each time $t$, if the state is $x(t) = [x_1(t), \ldots, x_N(t)]$, rank order $\{\gamma_i(x_i(t), \lambda_i, \alpha_i)\}$ in decreasing order (resolving any ties arbitrarily) and render the top $N_0$ active, the rest passive.

### III. The Whittle Dynamics

For fixed values of $x_{-i} := \{x_j, j \neq i\}$, the policy for the $i$th process is a threshold policy with threshold

$$T_i(x_{-i}, \Gamma) := \gamma_i^{-1}(\cdot, \lambda_i, \alpha_i) \left( \max_{j \neq i} \gamma_j(x_j, \lambda_j, \alpha_j) \right),$$

with $1 \leq i \leq N$. This will be monotone increasing in $x_{-i}$ by virtue of the structural properties established in [2]. The policy is captured by the step function

$$I\{i\text{th process active}\} = I\{u_i(n) = 1\} = I\{x_i \geq T_i(x_{-i}, \Gamma)\},$$

where $I\{\cdot\}$ is the indicator function. Following the approach of [16], [19], we approximate this function by

$$f_i(x_i, x_{-i}, \Gamma) := \frac{e^{\kappa_i(x_i - T_i(x_{-i}, \Gamma) - 0.5)}}{1 + e^{\kappa_i(x_i - T_i(x_{-i}, \Gamma) - 0.5)}}$$

where $\kappa_i > 0$. We treat this as the probability of picking 1, i.e., the active mode. The corresponding transition probabilities for the $i$th process are

$$p_i(y_i|x, \Gamma) = \cdots$$
We take a parametrized approximation for equation, i.e., the existence of solution of \( (\alpha_i x_i + u_i) \) interchangeably depending on the emphasis we want to put. Note that the actual operative policy is an index policy which deterministically picks at each time the \( N_0 \) sites to crawl. A probabilistic decision as above would violate the constraint. Thus the above is purely an approximation for analytic purposes. This is a common device in such learning schemes, employed so as to exploit the ease of optimization over continuous variables. If the scheme were a simulation-based off-line exercise, the probabilistic decision could be implemented, but this choice is not there for online schemes.

The transition probabilities are coupled, so the overall transition probability of the Markov chain is

\[
p([y_1, \ldots, y_N],[x_1, \ldots, x_N], \Gamma) := \prod_{i=1}^{N} p_i(y_i|x_i, x_{-i}),
\]

We take a parametrized approximation for \( T_i(x_{-i}, \Gamma) \) as \( T_i(x_{-i}, \Gamma) \approx \sum_{j=1}^{N} q_j^i(\Gamma) \psi_j(x_{-i}) \). From now on we use the \( q_j^i \)'s as surrogate parameters for \( \Gamma \) and suppress the \( \Gamma \)-dependence of \( q_j^i \)'s and other entities henceforth. Let \( U_i(x_i) := \{u_i, \alpha_i x_i + u_i\} \) and \( U(x) := \prod_i U_i(x_i) \). The Poisson equation for this Markov chain is

\[
V(x_1, \ldots, x_N) = \sum_{i=1}^{N} x_i - \beta + \sum_{y \in U} \prod_{i=1}^{N} p_i(y_i|x_i, x_{-i}) V(y_1, \ldots, y_N),
\]

where we have suppressed the dependence on the unknown parameter vector \( \Gamma \). Note that under Whittle policy, the system trajectory is periodic and traverses only a finite subset of the state space. For such a case, the well-posedness of the Poisson equation, i.e., the existence of solution \( (V(\cdot), \beta) \) with \( \beta \) uniquely characterized as \( \sum_{i=1}^{N} E_{x_i} [X_i(n)] \) and \( V(\cdot) \) unique up to an additive scalar, is easy to establish by a ‘vanishing discount’ argument. Arguing as in Proposition 1 of [10], we have, for

\[
q^i = [q_1^i, \ldots, q_N^i], q = [(q_1^T)^T : \ldots : (q_N^T)^T]^T, \quad \pi = \pi_\Gamma.
\]

\[
\nabla^q \beta = \int \pi(dx) \left( \sum_{y \in U(x)} \nabla^q p(y|x) V(y) \right)
= \int \pi(dx) \left( \sum_{y \in U(x)} p(y|x) \times \nabla^q \log p(y|x) V(y) \right),
\]

We use this as the basis of our learning scheme described in the next section. Explicit calculation leads to

\[
\nabla^q \log p(y|x) = \sum_{i=1}^{N} \nabla^q p_i(y_i|x) \frac{p_i(y_i|x)}{p_i(u_i|x)} \times \nabla^q \left( \frac{e^{\kappa_i(x_i - \sum_j q_j^i \psi_j(x_{-i}) - 0.5)}}{1 + e^{\kappa_i(x_i - \sum_j q_j^i \psi_j(x_{-i}) - 0.5)}} \right) + \frac{I\{y_i = u_i\}}{p_i(u_i|x)} \times \nabla^q \left( \frac{e^{\kappa_i(x_i - \sum_j q_j^i \psi_j(x_{-i}) - 0.5)}}{1 + e^{\kappa_i(x_i - \sum_j q_j^i \psi_j(x_{-i}) - 0.5)}} \right)
= \left[ \sum_{i=1}^{N} \frac{I\{y_i = \alpha_i x_i + u_i\}}{p_i(\alpha_i x_i + u_i)} \nabla^q (1 - p_i(y_i|x)) \right],
\]

where we have exhibited the partial derivative w.r.t. \( q_j^i \).

IV. THE LEARNING ALGORITHM

Let \( V(x) \approx \sum_{m=1}^{\ell} r(m) \phi_m(x) \) where the \( \phi_m \)'s are pre-selected features (e.g., polynomials). Let \( \phi(x) := [\phi_1(x), \ldots, \phi_\ell(x)]^T \). We adapt the LSPE(0) algorithm from [23] as follows.
The parameter vector \( r := [r_1, \ldots, r_q]^T \) is updated according to: for \( t \geq 0, 
\)
\[
r(n + 1) = r(n) + \gamma B^{-1}(n)(\bar{A}(n)r(n) + \bar{b}(n)),
\]
\[
\eta(n + 1) = \eta(n) + \frac{1}{n+1} \times \left( \sum_i X_i(n) - \eta(n) \right)
\]
\[
\bar{B}(n) = B(n - 1) + \frac{1}{n+1} \times [\phi(X(n))\phi(X(n))^T - B(n - 1)],
\]
\[
\bar{B}(-1) = \epsilon I,
\]
\[
\bar{A}(n) = \bar{A}(n - 1) + \frac{1}{n+1} \times \left( \phi(X(n))\phi(X(n + 1))^T - \phi(X(n))^T - \bar{A}(n - 1) \right),
\]
\[
\bar{b}(n) = \bar{b}(n - 1) + \frac{1}{n+1} \times \left( \phi(X(n)) \left( \sum_i X_i(n) - \eta(n) \right) - \bar{b}(n - 1) \right),
\]

and finally,
\[
\bar{B}^{-1}(n) = \frac{n}{n-1} \left( \bar{B}^{-1}(n - 1) - \frac{\bar{B}^{-1}(n)(\phi(X(n))\phi(X(n))^T \bar{B}^{-1}(n - 1) - \phi(X(n))^T \bar{B}^{-1}(n - 1)\phi(n))}{n-1 + \phi(X(n))^T \bar{B}^{-1}(n - 1)\phi(n)} \right).
\]

The last iterate is simply the Sherman-Morrison formula to update the inverse. This is coupled with the stochastic gradient ascent for \( \{q(n)\} \) based on (7) as follows:
\[
q(n + 1) = q(n) + a(n)\nabla q \log p(X(n + 1)|X(n)) \times \left( \sum_{i=1}^{t} r_i(n)\phi_i(X(n+1)) \right),
\]

where \( a(n) > 0 \) satisfy
\[
\sum_{n} a(n) = \infty, \quad \sum_{n} a(n)^2 < \infty.
\]

The term \( \nabla q \log p(X(n + 1)|X(n)) \) is given by (8). In the iterates for \( \eta(n), \bar{B}(n), \bar{A}(n) \), we can replace the stepsize \( 1/(n+1) \) by \( b(n) \) satisfying
\[
\sum_{n} b(n) = \infty, \quad \sum_{n} b(n)^2 < \infty, \quad a(n) = o(b(n)).
\]

An example is: for some \( c \in (0, 1) \),
\[
a(n) = \frac{c}{n}, \quad b(n) = \frac{c}{n^{1.2}}.
\]

V. Numerical Experiments

We compare the performance of the proposed learning scheme with the performance of the Whittle index. For the comparison purposes we take the same numerical example as in [2]. The example has 4 arms and the parameters are given in Table I.

<table>
<thead>
<tr>
<th>( l )</th>
<th>( \lambda_1 )</th>
<th>( \lambda_2 )</th>
<th>( \lambda_3 )</th>
<th>( \lambda_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>250</td>
<td>250</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>2</td>
<td>0.7</td>
<td>0.35</td>
<td>0.7</td>
<td>0.21</td>
</tr>
<tr>
<td>( M )</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
</tbody>
</table>

The other parameters of the algorithm are as follows: \( \kappa = 2 \) and \( \gamma = 0.05 \). As approximating functions, we take:
\[
T_i(x_{-i}) = q_0 + q_i \sum_{k \neq i} x_k + q_i^1 \sum_{k \neq i} x_k^2,
\]
\[
\phi = [x_1 \cdots x_N \ x_1^2 \ x_1x_2 \cdots x_{N-1}x_N \ x_N^2].
\]

While running the initial version of the algorithm we noticed that the algorithm quickly runs into numerical instability. We have identified two possible reasons for the instability. The first is that the value of the thresholds could reach small or large values, which poses numerical problems in evaluating the exponents of the logit distribution. The second issue, which probably comes from the deterministic nature of the original model, is that the matrix \( B \) becomes close to rank-deficient. To overcome the first issue, we project the value of the thresholds on the intervals \( [u_i, u_i/(1-\alpha_i)] \).

Note that according to the model definition, the state variables cannot go outside the intervals
Inversion, \( B^{-1} \), thus this is a legitimate step. To overcome the second issue, we replaced the matrix inversion \( B^{-1} \) with its Tikhonov regularization \( (B^T B + \delta I)^{-1} B^T \) for a small \( \delta > 0 \). These practical adjustments helped solve the numerical problems.

We have run the algorithm for \( N = 10000 \) steps. Note that the choice of \( \kappa_i \) controls the average value and dispersion of the number of engaged arms. With our choice \( \kappa_i = 2 \), we have on average one arm sampled and obtain the reward around 160, which is a value between the rewards that we could get by sampling constantly either the best or the second best arm. It is significantly smaller than what we can get by using Whittle index with one arm, viz., 283 [2]. On the other hand, our algorithm hardly uses any information about the system parameters, whereas the complete knowledge of the system parameters is needed when applying the Whittle index as in [2].

**ACKNOWLEDGMENT**

The authors were supported in part by the INRIA-DST project ‘Machine Learning for Network Analytics’ administered by the Indo-French Centre for Promotion of Advanced Research (IFCPAR). VB was also supported by a J. C. Bose Fellowship from the Government of India. The work of KA is also supported by the PIA ANSWER project PIA FSN2 no.P159564-2661789/DOS0060094 between Inria and Qwant.

**REFERENCES**


[20] Lefortier, D.; Ostroumova, L.; Samosvat, E. and

