Spectral MAB for Unknown Graph Processes

Laura Toni[1] and Pascal Frossard[2]
[1] Dept. of Electronic and Electrical Engineering, UCL, London, UK
[2] Signal Processing Laboratory (LTS4), EPFL, Switzerland

***Abstract*—In this work, we study graph-based multi-arms bandit (MAB) problems aimed at optimizing actions on irregular and high-dimensional graphs. More formally, we consider a decision-maker that takes sequential actions over time and observes the experienced reward, defined as a function of a sparse graph signal. The goal is to optimize the action policy, which maximizes the reward experienced over time. The main challenges are represented by the system uncertainty (*i.e.*, unknown parameters of the sparse graph signal model) and the high-dimensional search space. The uncertainty can be faced by online learning strategies that infer the system dynamics while taking the appropriate actions. However, the high-dimensionality makes online learning strategies highly inefficient. To overcome this limitation, we propose a novel graph-based MAB algorithm, which is data-efficient also in high-dimensional systems. The key intuition is to infer the nature of the graph processes by learning in the graph-spectral domain, and exploit this knowledge while optimizing the actions. In particular, we model the graph signal with a sparse dictionary-based representation and we propose an online sequential decision strategy that learns the parameters of the graph processes while optimizing the action strategy.**

I. INTRODUCTION

We are surrounded by large-scale interconnected systems (transportation networks, social networks, etc.), which create services and produce massive amounts of data. This has pushed researchers in developing *understanding* for high-dimensional data networks. However, it is also essential to focus on *controlling and optimizing* these data networks. For example, advertisements agencies might be in need to place ads properly among users to maximize their profit within an entire social networks, see Fig. 1. Also, a sparse set of energy sources (or cooling sources) might need to be placed within grid networks (or cooling systems). These are examples of decision making strategies (DMSs) over high-dimensional and irregular networks. When the dynamic of the system to control is not known a priori, machine learning provides us with online DMSs to learn the dynamics while controlling the system.

Multi-arms bandit problems are very common online DMSs and, by now, they are well-understood for small strategy sets [1]. In the classical stochastic k-armed bandit problem, a decision maker chooses one out of the k possible arms (actions) and experiences an instantaneous reward, which is chosen from an unknown distribution associated with that arm. The goal is to learn from the experience (by trial-and-error) the arm with the best distribution, *i.e.*, the arm with the highest reward on average. The performance of these learning strategies is measured in terms of regret, which is the difference between the reward incurred by the algorithm and the optimal reward. However, the regret scales with the ambient dimension, either linearly or as a square root [1], which makes the problem intractable in scenarios with infinitely large strategy sets, as large-scale network optimizations, recommendation systems etc. To learn efficiently in complex domains data (data-efficient machine learning), one must ultimately be able to exploit the structure of the high-dimensional ambient space [2], [3]. However, properly exploiting the structure in irregular and high-dimensional networks is an open challenge. In this work, we address this challenge by applying the tenets of graph signal processing (able to model non-Euclidean large systems) to online DMSs. In particular, we develop novel graph-based multi-arm bandit problems as sequential decision strategies for high-dimensional networks. We claim that looking at the arms as nodes on the graph and at the payoff as signal on the graph allows to infer and exploit the geometry of the ambient space to deduce the process driving the signal on the graph, *i.e.*, the reward. The key intuition is to infer the problem structure by learning in the (sparse) graph-spectral domain, and exploit this knowledge while optimizing the actions in the (high-dimensional) vertex domain. Fig. 2 depicts the main differences between a classical MAB problem and the proposed graph-based MAB one.

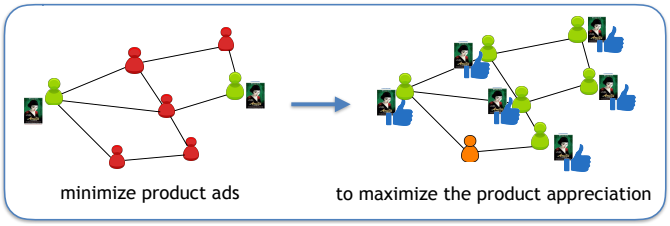
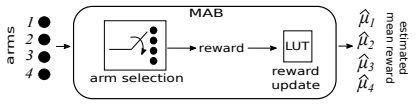


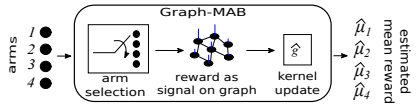
Figure 1. Applicative example of ads placement within a social networks.

In this work, we assume a general representation of processes taking place on large networks and we do not limit ourselves to smooth-reward assumption [4]. We consider a network (that we model as graph) and a decision maker optimizing the placement of sparse sources on the network. Following an unknown graph process, the source signal spreads across the network over time, leading to a resulting signal that characterizes the instantaneous reward. While the graph structure is assumed to be known, the decision-maker needs to learn over time the model driving the graph process to be able to take the optimal decision on the source placement. By taking sequential decisions, a training set is built and used as input to a parametric dictionary learning algorithm to infer kernels that sparsely represent graph signals. Similarly to [5],

ISBN 978-90-827970-1-5 © EURASIP 2018



(a) MAB



(b) Graph-based MAB

Figure 2. Graphical visulazation of both the classical MAB and the graph-based MAB problems.

the graph process is approximated by a smooth polynomial function, which incorporates the graph Laplacian (*i.e.*, it takes into account the graph structure). The key intuition is to learn the sparse coefficient vector of the polynomial function to characterize the graph signal resulting from any originating source (being then able to optimize the source signal). This allows us to find the best tradeoff between exploitation (optimized based on the current knowledge of the system) and exploration (suboptimal actions that might reveal unknown behaviors of the system) despite the large dimensionality of the problem. We then characterize the confidence bound of the MAB learner as a function of the graph structure and use it to propose a learning algorithm. Simulation results show accuracy of the our graph-MAB strategy when compared to baseline ones. Finally, we also provide a study on the graph topology that favors a faster learning process. We show that more connected graphs and more sparse source signals leads to a more accurate estimation of the graph process.

II. Sequential Decision Strategies over Graphs

In the following, we formulate the problem we aim to solve on high-dimensional graphs. We then provide an overview of the classical MAB problems. Finally, we show that our problem can be formulated as a classical MAB problem, that however leads to a data-inefficient learning method.

A. Problem Formulation

We consider a weighted and undirected graph $G = (\mathcal{V}, \mathcal{E}, W)$ (known to the decision maker), with $\mathcal{V}$ being the vertex set with $|\mathcal{V}| = N$, $\mathcal{E}$ the edge sets, and W the $N \times N$ adjacency matrix. Let $\boldsymbol{h}_t = [h_{1,t}, h_{2,t}, \dots, h_{N,t}]^T$, be the source signal over G (*i.e.*, action taken by the decision maker), with $h_{n,t} \in [0, 1]$. The source signal will spread across G over time following an *unknown* process, and the resulting signal after a time period of Δ can be expressed as function of the source signal $\boldsymbol{h}_t$ as

$$\boldsymbol{y}_t = \boldsymbol{\mathcal{D}}\boldsymbol{h}_t + \boldsymbol{\epsilon}_t$$

where $\boldsymbol{y}_t = [y_{1,t}, \dots, y_{N,t}]^T$, and $\boldsymbol{\epsilon}_t = [\epsilon_{1,t}, \dots, \epsilon_{N,t}]^T$ is a random noise. In the applicative example of ads placed over a social network, G represents the social network, $\boldsymbol{h}_t$ the location of ads placed at t, and $y_{n,t}$ the resulting popularity of the content of interest at the vertex location n at time $t + \Delta$. The resulting signal $\boldsymbol{y}_t$ is generated by an underpinning process, which is unknown to the decision maker. Given $\boldsymbol{y}_t$, the instantaneous reward of the decision maker is denoted by $r(\boldsymbol{h}_t) = \boldsymbol{M}\boldsymbol{y}_t$, where $\boldsymbol{M}$ is a $1 \times N$ binary matrix, showing the nodes over which the signal is measured to build the reward. Thus, the reward is a mean of the signal $\boldsymbol{y}_t$ at the location activated by the masking matrix $\boldsymbol{M}$[1].

The mean reward experienced by the decision maker when taking action $\boldsymbol{h}_t$ is given by $\mu(\boldsymbol{h}_t) = \mathbb{E}\{r(\boldsymbol{h}_t)\} = \boldsymbol{M}\boldsymbol{\mathcal{D}}\boldsymbol{h}_t$. Let consider sequential decision strategies, in which the decision maker takes actions over time. At the decision opportunity t, the action $\boldsymbol{h}_t$ is selected and the reward $r(\boldsymbol{h}_t)$ is observed. We assume that the rewards associated to consecutive actions (*e.g.*, $r(\boldsymbol{h}_t)$ and $r(\boldsymbol{h}_{t'})$) are i.i.d.. In the example of ads placement, different ads content are placed over time and therefore the associated appreciation of the user does not depend on the ads placed previously in time. Let denote by $\mathcal{A}$ the set of feasible actions that the decision maker can take, defined as

$$\mathcal{A} = \{\boldsymbol{h} \mid ||\boldsymbol{h}||_0 \leq T_0 \ \wedge \ h_n \in [0, 1], \ n = 1, ..., N\}$$

where T_0 is the maximum sparsity level of the actions. In an ideal case in which the mean reward $\mu(\boldsymbol{h})$ is known for all actions, the optimal action $\mu^\star = \max_{\boldsymbol{h} \in \mathcal{A}} \mu(\boldsymbol{h})$ would be selected at each decision opportunity. In practice, the mean reward is not known and the decision maker needs to learn from past actions the best policy to take such that the reward of future actions is maximized. This translates in the following optimization problem

$$\max_{\{\boldsymbol{h}_t\}_t} \mathbb{E}\left\{\sum_{t=1}^{T} r(\boldsymbol{h}_t)\right\} \quad \text{s.t.} \quad \boldsymbol{h}_t \in \mathcal{A} \tag{1}$$

The faster the decision maker learns the mapping between actions and mean reward, the more profitable will be the future actions.

B. The MAB problem

We now provide the classical MAB problem formulation as shown in [1] and references therein. Let $a \in \mathcal{A}$ be one possible action and $\mathcal{A}$ the set of feasible actions. At time t, the learner selects the action a and experiences an instant payoff $r_{a,t}$, which is drawn from a stochastic distribution with mean value μ_a. Being the mean payoff unknown *a priori*, the learner does not select the optimal arm (*i.e.*, the one that maximizes μ_a) immediately. It rather selects suboptimal actions, leading to a cumulative regret after T decisions defined as $R(T) = T\mu^\star - \sum_{t=1}^{T} r_{a_t,t}$, being a_t the arm selected at the t-th decision opportunity and $\mu^\star$ the mean reward of the optimal arm.

The classical algorithm for MAB problems is the UCB (or its improved versions, such as UCB-Tuned) [6]. The key idea is that at each round the arm with the highest bandit index is

[1]In general, the reward can be any a function of the resulting signal such as the mean of $\boldsymbol{y}$ or $||\boldsymbol{y}^* - \boldsymbol{y}||_2$, with $\boldsymbol{y}^*$ being the target signal.

ISBN 978-90-827970-1-5 © EURASIP 2018

selected. In the classical UCB algorithm, the bandit index of arm a at time t is

$$b_{a,t} = \hat{\mu}_{a_t} + \sqrt{\frac{2\log t}{N_{a,t}}} \tag{2}$$

with $N_{a,t}$ being the number of times the arm a has been selected up to t, and $\hat{\mu}_{a,t}$ is the mean reward for arm a estimated at the tth decision opportunity.

C. Graph optimization as MAB problem

We observe that the maximization of the reward in (1) is equivalent to the following minimization

$$\min_{\{\boldsymbol{h}_t\}_t \in \mathcal{A}} T\mu^\star - \mathbb{E}\left\{\sum_{t=1}^{T} r_t(\boldsymbol{h})\right\}. \tag{3}$$

At the same time, we notice that the above optimization problem is the minimization of the mean cumulative regret of MAB problems in which each action a is a source signal $\boldsymbol{h}$. Adopting UCB or similar solvers for MABs, the regret is asymptotically minimized [1], therefore the reward is asymptotically maximized. This means that solving (3) as MAB problems is asymptotically equivalent to solving (1).

The UCB algorithm achieves a sublinear regret, *i.e.*, $R(t) = \mathcal{O}(|\mathcal{A}|\log t)$. In particular, the regret of the learning algorithm scales with the ambient dimension. In our case, the action space (or the total number of possible arms) is $A = \binom{N}{T_o}$, if T_0 is the imposed sparsity of $\boldsymbol{h}$. It is easy to understand that this action space is not sustainable for current MAB problems. By way of example, in a network with 100 nodes, actions with sparsity $T_0 \leq 10$ would lead to an action space with cardinality $|\mathcal{A}| \geq 10^{13}$.

III. Proposed Algorithm

A. Learning in the spectral domain

We now overcome the above limitation by taking into account the graph structure in the sequential optimization process, in particular we show the gain in learning structured dictionaries that sparsely represent the signal on the graph and thus permit to infer the nature of the graph processes. Then acting/optimizing in the high-dimensional domain of the actions is made easier since the graph process is well estimated. We start from the assumption that graph signals can be modeled as combinations of overlapping local patterns or generating kernel $\hat{g}(\cdot)$, that are a function of the eigenvalues of the Laplacian and characterize the graph pattern in the spectral domain [5]. In more details, let define the dictionary $\boldsymbol{\mathcal{D}}$ as $\boldsymbol{\mathcal{D}} = \hat{g}(L) = \sum_{k=0}^{K} \alpha_k L^k$, where the last equality holds in the case of smooth kernel functions. This means that we impose that the kernel translated in the vertex n has a support contained in a ball of K hops from vertex n. Under the assumption of smoothness of the kernels, the reward associated to $\boldsymbol{h}$ is expressed as linear combinations of coefficients that define the dictionaries:

$$y_{n,t} = \sum_{m=1}^{N} h_{m,t} \sum_{k=0}^{K} \alpha_k (L^k)_{n,m} + \epsilon_{n,t} \tag{4}$$

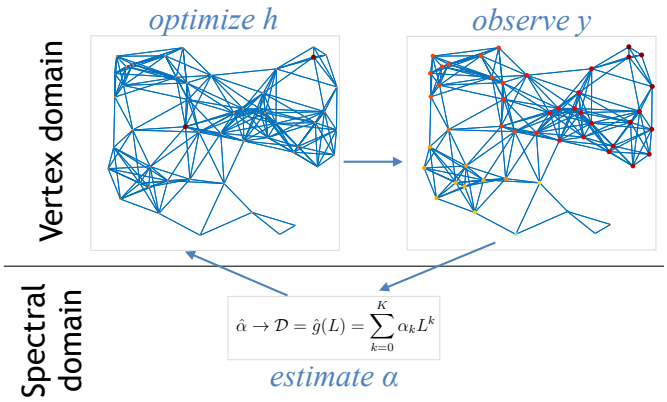


Figure 3. Graphical representation of the kernel UCB.

where $(L^k)_{n,m}$ is the (m,n) entry of L^k. Finally, $\boldsymbol{\epsilon}_t = [\epsilon_{1,t}, \epsilon_{2,t}, \ldots, \epsilon_{N,t}]^T$ is a Gaussian and N-dimensional random variable with $\epsilon_{n,t} \sim \mathcal{N}(0, \sigma_e^2)$. With the following matrix notations $\boldsymbol{P} = [L^0, L^1, L^2, \ldots, L^K]$, with $P \in \mathbb{R}^{N \times N(K+1)}$, and $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, \ldots, \alpha_K]^T$, we can then express the resulting signal $\boldsymbol{y}_t$ as

$$\boldsymbol{y}_t = \boldsymbol{P}\boldsymbol{I}_{K+1} \otimes \boldsymbol{h}_t \boldsymbol{\alpha} = \boldsymbol{P}\boldsymbol{H}_t \boldsymbol{\alpha} \tag{5}$$

with $\boldsymbol{I}_{K+1}$ being the $(K+1) \times (K+1)$ identity matrix, and $\boldsymbol{H}_t = \boldsymbol{I}_{K+1} \otimes \boldsymbol{h}_t$, with $\boldsymbol{H}_t \in \mathbb{R}^{N(K+1) \times (K+1)}$.

At the decision opportunity t, the decision-maker takes a given action $\boldsymbol{h}_t$ on the graph G, leading to an instantaneous reward of

$$r(\boldsymbol{h}_t) = \boldsymbol{M}\boldsymbol{y}_t = \boldsymbol{M}\boldsymbol{P}\boldsymbol{H}_t\boldsymbol{\alpha} + \boldsymbol{M}\boldsymbol{\epsilon}_t = \boldsymbol{X}_t\boldsymbol{\alpha} + \boldsymbol{\eta}_t \tag{6}$$

where $\boldsymbol{\eta}_t = [\eta_1, \ldots, \eta_N]^T = \boldsymbol{M}\boldsymbol{\epsilon}$, with $\eta_n \sim \mathcal{N}(0, N\sigma_e^2)$, and $\boldsymbol{X}_t = \boldsymbol{M}\boldsymbol{P}\boldsymbol{H}_t$, with $\boldsymbol{X}_t \in \mathbb{R}^{N \times (K+1)}$. This means that the reward can be expressed as a linear combination of the K-degree polynomial $\boldsymbol{\alpha}$ and the matrix $\boldsymbol{X}_t$, which includes both the graph structure information (via the Laplacian L) and the action $\boldsymbol{h}$. In the following, we show how to exploit this linear combination to learn the polynomial $\boldsymbol{\alpha}$ and optimize the sequential decision strategy of the decision maker.

B. Kernel UCB

We now propose a novel algorithm that learns in the spectral domain and then acts in the vertex domain. The algorithm is composed of two steps: 1) refinement of the coefficients estimate, 2) selection of the arm given the updated knowledge of the system, as graphically shown in Fig. 3.

Step 1: Coefficients estimation

Let consider the t-th decision opportunity, when $t-1$ decisions have been already taken and the corresponding signal and reward have been observed. The training set built over time corresponds to $\{(\boldsymbol{h}_\tau, \boldsymbol{y}_\tau)\}_{\tau=1}^{t-1}$, where we recall that the randomness is due to the random noise $\boldsymbol{\epsilon}_\tau$ and $p(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{\alpha}) \sim \mathcal{N}(\mathcal{D}\boldsymbol{h}, \sigma_e^2 \boldsymbol{I}_N)$. For large t, maximizing the MAP probability

ISBN 978-90-827970-1-5 © EURASIP 2018

$p(\boldsymbol{\alpha}|\boldsymbol{y},\boldsymbol{h})$ corresponds to minimize the l_2-regularized least-square estimate of $\boldsymbol{\alpha}$, which leads to the following estimation problem:

$$\hat{\boldsymbol{\alpha}}_t : \arg\min_{\boldsymbol{\alpha}} \sum_{\tau=1}^{t-1} ||\boldsymbol{P}\boldsymbol{H}_{\boldsymbol{\tau}}\boldsymbol{\alpha} - \boldsymbol{y}_\tau||_2^2 + \lambda||\boldsymbol{\alpha}||_2^2 . \quad (7)$$

Thus $\hat{\boldsymbol{\alpha}}_t = \left[\sum_{\tau=1}^{t-1} \boldsymbol{Z}_\tau^T \boldsymbol{Z}_\tau + \lambda \boldsymbol{I}_{K+1}\right]^{-1} \sum_{\tau=1}^{t-1} \boldsymbol{Z}_\tau^T y_\tau = \boldsymbol{V}_t^{-1}\boldsymbol{Z}_{1:t}^T\boldsymbol{Y}_t$, with $\boldsymbol{Z}_{1:t} = [\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_{t-1}]^T$, $\boldsymbol{Z}_\tau = \boldsymbol{P}\boldsymbol{H}_\tau$, $\boldsymbol{Y}_t = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{t-1}]^T$, and $\boldsymbol{V}_t = \boldsymbol{Z}_{1:t}^T\boldsymbol{Z}_{1:t} + \lambda\boldsymbol{I}_{K+1}$.

In practice, since the training set is built over time, it is a small set to begin with. Therefore the l^2-regularized least-square estimate in (7) leads to an approximation of the actual polynomial $\boldsymbol{\alpha}$, and this approximated estimate is refined at each decision opportunity.

Step 2: Action selection

Once the estimation of the $\boldsymbol{\alpha}$ coefficients is refined, the decision maker needs to select the action to take at the t-th decision opportunity. Following the theory of linear UCB [7], the decision maker selects the action $\boldsymbol{h}$ (and therefore $\boldsymbol{X} = \boldsymbol{M}\boldsymbol{P}\boldsymbol{I}_{K+1} \otimes \boldsymbol{h}$) such that

$$\boldsymbol{h}_t : \arg\max_{\boldsymbol{h}\in\mathcal{A}} \max_{\alpha\in E_t} \boldsymbol{X}\boldsymbol{\alpha} \quad (8)$$

where the confidence bound E_t is an ellipsoid centered in $\hat{\boldsymbol{\alpha}}_t$ defined such that $\boldsymbol{\alpha}_* \in E_t$ with probability $1-\delta$ for all $t \geq 1$. With a confidence bound E_t such that $E_t : \{||\hat{\boldsymbol{\alpha}}_t - \boldsymbol{\alpha}_*|| \leq c_t\}$, with $c_t = R\left[\sqrt{K\log(1+tNT_0d/\lambda)} + \sqrt{2\log 1/\delta}\right] + \lambda^{1/2}S$, the maximization in (8) becomes

$$\begin{aligned}\boldsymbol{h}_t &= \arg\max_{\boldsymbol{h}\in\mathcal{A}} \max_{\alpha\in E_t} \boldsymbol{X}\boldsymbol{\alpha} = \arg\max_{\boldsymbol{h}\in\mathcal{A}} \boldsymbol{X}\boldsymbol{\alpha} + c_t\sqrt{\boldsymbol{X}\boldsymbol{V}_t^{-1}\boldsymbol{X}^T} \\ &= \arg\max_{\boldsymbol{h}\in\mathcal{A}} \left[\boldsymbol{M}\boldsymbol{P}\boldsymbol{H}\hat{\boldsymbol{\alpha}}_t + c_t||\boldsymbol{M}\boldsymbol{P}\boldsymbol{H}||_{\boldsymbol{V}_t^{-1}}\right] . \quad (9)\end{aligned}$$

Algorithm 1 Kernel-UCB

Input:

N: number of nodes, T_0: sparsity level of initial signal $\boldsymbol{h}$,
K: sparsity of the basis coefficients, $\boldsymbol{M}$: reward mask.
λ, δ: regularization and confidence parameters
R, S: upper bounds on the noise and $\boldsymbol{\alpha}_*$
$t = 1$
while $t \leq T$ **do**
  Refine estimate of the coefficients
  $\boldsymbol{X}_{1:t} = [\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{t-1}]^T$
  $\boldsymbol{Y}_{1:t} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{t-1}]^T$
  $\boldsymbol{V}_t = \boldsymbol{X}_{1:t}^T\boldsymbol{X}_{1:t} + \lambda\boldsymbol{I}_{K+1}$
  $\hat{\boldsymbol{\alpha}}_t = \boldsymbol{V}_t^{-1}\boldsymbol{X}_{1:t}^T\boldsymbol{Y}_{1:t}$
  Evaluate the confidence bound and select the best action
  $c_t = R\left[\sqrt{K\log(1+tNT_0d/\lambda)} + \sqrt{2\log 1/\delta}\right] + \lambda^{1/2}S$
  $\boldsymbol{h}_t : \arg\max_{\boldsymbol{h}\in\mathcal{A}} \left[\boldsymbol{M}\boldsymbol{P}\boldsymbol{H}\hat{\boldsymbol{\alpha}}_t + c_t||\boldsymbol{M}\boldsymbol{P}\boldsymbol{H}||_{\boldsymbol{V}_t^{-1}}\right]$
  Observe the resulting signal $\boldsymbol{y}_t$ and the instantaneous reward $r(\boldsymbol{y}_t)$
  $t = t + 1$
end while

This optimization characterizes the Step 2, *i.e.*, the action selection. The above problem maximizes a convex objective function over a polytope[2], defined by the two constraints. It can be shown that, if the objective function has a maximum value on the feasible region then it is at the edges of the polytope. Therefore, the problem reduces to a finite computation of the objective function over the finite number of extreme points.

In Algorithm 1, we summarize the main steps of the proposed Kernel-UCB strategy.

IV. Results

As benchmark solution, we propose an algorithm that first builds a training set (in the first T_L decision strategies), then estimates the generating kernel and selects the best arm. In the first T_L decision opportunities randomly selected action are taken. We label this algorithm AAL (act after learning). We then provide results for the proposed Kernel-UCB algorithm with and without confidence bound, *i.e.*, c_t as evaluated from Algorithm 1, or $c_t = 0$. When $c_t = 0$, the algorithm selects the future actions only based on the current estimate of the system dynamics, neglecting uncertainty on this estimate .

We carry our experiments on radial basis function (RBF) random graphs, where we generate the coordinates of the vertices uniformly at random in the unit square, and we set the edge weights based on a thresholded Gaussian kernel function so that $W(i,j) = \exp(-[dist(i,j)^2)/2\sigma)$ if the distance between vertices i and j is less than or equal to T, and zero otherwise. We further set $\sigma = 0.5$ and we vary T to construct graphs which are more or less densely connected. We then consider that each signal on the graph is characterized by the originating signal $\boldsymbol{h}$, the generating kernel and an additive random noise ϵ_t with zero mean and variance σ_e^2. This leads to a R value in the spectral UCB of $R = \sigma_e$. Remaining parameters of the sequential decision strategy are the following: $\lambda = 0.01$, $\delta = 0.01$.

We now show its performance with respect to the AAL baseline algorithm, for a randomly generated graph (RBF model) with $N = 100$, and sparsity level $T_0 = 4$. The mask $\boldsymbol{M}$ is randomly generated and it covers 20% of the nodes. Fig. 4 depicts the cumulative regret as a function of the time (in terms of decision opportunities) for the considered graph. Each point is averaged over 100 realizations (when at each realization both the graph and the noise of the signal on graph is generated). The Kernel-UCB is compared to the baseline algorithm AAL with different learning time T_L, namely $T_L = 10$ and $T_L = 20$. Note the longer is the learning time, the better is the estimate of the polynomial $\boldsymbol{\alpha}$. However, since during the learning phase actions are selected at random, the longer is also the suboptimal phase. From Fig. 4, we observe that the Kernel-UCB (both with and without confidence bound) outperforms the baseline algorithm. We

[2]For the sake of brevity, we skip the proof of convexity. The intuition is that $\boldsymbol{V}_t^{-1}$ can be decomposed into $\boldsymbol{V}_t^{-1} = \boldsymbol{L}^T\boldsymbol{L}$ and the objective function can be expressed as a sum of an affine and l_2-norm term.

ISBN 978-90-827970-1-5 © EURASIP 2018

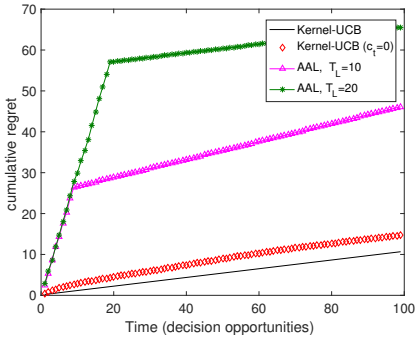


Figure 4. Cumulative regret vs. time for randomly generated graphs with $N = 100$, diffusion process (with $\tau = 0.5$) and sparsity level $T_0 = 5$.

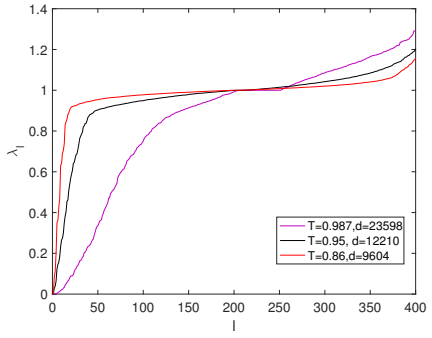
also notice that the baseline with $T_L = 10$ leads to a non-accurate estimate of the polynomial $\boldsymbol{\alpha}$. Therefore, after the learning phase the decision-maker selects future actions under the assumption of a wrong estimated $\boldsymbol{\alpha}$. This leads to a large level of suboptimality and therefore to a rapidly increasing cumulative regret. Finally, as expected, the proposed algorithm with the confidence bound take into consideration outperforms the one with $c_t = 0$.

We are now interested in understanding how much the graph topology correlates to the performance of the learning process. To do this, we first consider a randomly generated training set of 300 signals, and we then estimate the accuracy of the learned polynomial α. Implicitly, the better the estimate, the more efficient the decision maker. To measure the accuracy of the estimate, we evaluate the error on the resulting signal given the action $\boldsymbol{h}$ of test signals. Basically we evaluate $(1/|Y_{Test}|)\sum_i ||\boldsymbol{y}_i - \mathcal{D}\boldsymbol{h}_i||_2^2/||\boldsymbol{y}_i||^2$, where $|Y_{Test}|$ is the cardinality of the testing set.

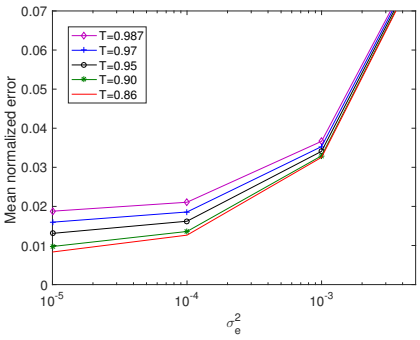
We are interested in studying how much the estimation error depends on the connectivity of the graph, and therefore on the Laplacian. Let us consider graphs generated with the RBF model again. By changing the threshold parameter T, we generate more or less densely connected graph. Higher levels of connectivity also leads to a different profiles of the eigenvalues of the Laplacian λ_l, as observed from Fig. 5(a), where the values of λ_l are provided for different graph topologies. In particular, we provide λ_l for graph topologies with $N = 400$ and $T = 0.987, 0.95$ and 0.86. In the legend, we also provide the power sum of the eigenvalues, namely $d = \sum_{k=0}^{K}\sum_{l=1}^{N}\lambda_l^k$, for each graph topology. As a consequence, more connected graphs lead to a more accurate estimate of the generating kernels, see Fig. 5(b). The intuition is that a more densely connected graph leads to a more informative resulting signal $\boldsymbol{y}$ and therefore to a better estimate. Mathematically, this can also be deduced by observing the distribution of the Laplacian eigenvalues (Fig. 5(a)) and the associated power sum d.

V. Conclusions

We proposed graph-based online sequential strategies, which are data-efficient in high-dimensional problems. The



(a) Graph Laplacian Eigenvalues



(b) Estimation Error

Figure 5. Graph Laplacian distribution and signal estimation error for random graphs with different levels of connectivity, $N = 400$ nodes, and sparsity value $T_0 = 15$..

key intuition is to infer the problem structure by learning in the (low-dimensional) graph-spectral domain, and exploit this knowledge while optimizing the actions in the (high-dimensional) vertex domain. This allows us to find the best tradeoff between exploitation and exploration despite the large dimensionality of the problem. We analyse the system performance also in correlation with the graph connectivity. Simulation results show the gain of proposed algorithm when compared to baseline ones.

References

[1] S. Bubeck, N. Cesa-Bianchi *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[2] Y. Y. Jia and S. Mannor, "Unimodal bandits," in *Proc. International Conference on Machine Learning (ICML)*, 2011, pp. 41–48.

[3] A. Slivkins, "Contextual bandits with similarity information." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2533–2568, 2014.

[4] M. Valko, R. Munos, B. Kveton, and T. Kocak, "Spectral bandits for smooth graph functions." in *Proc. International Conference on Machine Learning (ICML)*, 2014.

[5] D. Thanou, D. I. Shuman, and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Trans. on Signal Processing*, vol. 62, no. 15, pp. 3849–3862, 2014.

[6] A. Garivier and O. Cappé, "The KL-UCB algorithm for bounded stochastic bandits and beyond." in *Proc. Computational Learning Theory (COLT)*, 2011, pp. 359–376.

[7] W. Chu, L. Li, L. Reyzin, and R. E. Schapire, "Contextual bandits with linear payoff functions." in *Proc. Artificial Intelligence and Statistics Conference (AISTATS)*, vol. 15, 2011, pp. 208–214.

ISBN 978-90-827970-1-5 © EURASIP 2018