Online Network Source Optimization with Graph-Kernel MAB

Laura Toni[**1] and Pascal Frossard[2]

[1] EEE Dept, University College London, London, United Kingdom
[2] LTS4, EPFL, Lausanne, Switzerland

Abstract. In this work, we study the problem of learning online the optimal source placement in networks, such that the reward obtained from a priori unknown network processes is maximized. We consider graph-based multi-arms bandit problems aimed at optimizing actions on high-dimensional networks, with a decision-maker that takes sequential actions over time and observes the resulting reward on the network. The goal is to optimize the action policy in order to maximize the reward experienced over time. The main challenges are represented by the uncertainty about the system and the properties of the network processes, as well as the high-dimensional search space in large networks. The uncertainty can be addressed by online learning strategies that infer the system behavior from past experience. However, the action-reward mapping is typically learned with a sublinear regret (i.e., suboptimality in terms of reward gap), which increases with the dimension of the search space, leading to highly suboptimal solutions in large networks. To overcome this limitation, we describe the network processes with an adaptive graph dictionary model, which leads to sparse spectral representations. This enables a data-efficient learning framework, whose learning rate scales the dimension of the spectral representation model instead of the one of the network. We then propose an online sequential decision strategy that learns the parameters of the spectral representation while optimizing the action strategy. We derive the performance guarantees that depend on network parameters and show the correlation between the network topology and the learning curve of the sequential decision strategy. Simulations are then carried out in source placement tasks where the proposed online learning algorithm outperforms baseline offline methods that typically separate the learning phase from the testing one. The results confirm the theoretical findings, and further highlight the gain of the proposed online learning strategy in terms of cumulative regret, sample efficiency and computational complexity.

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

Large-scale interconnected systems (transportation networks, social networks, etc.), which create services and produce massive amounts of data, are becom-

[**] Corresponding author: l.toni@ucl.ac.uk

ing predominant in many application domains. The management of such networked systems is exceedingly hard because of their intrinsic and constantly growing complexity. Many works have been proposed to tackle this problem (e.g., model based optimal control, consensus works [25, 26, 31, 45, 46], Bayesian approaches [2], etc.) but with a limited focus on online learning and control of large-scale networks. The latter becomes extremely challenging with highly dynamic and high dimensional network processes that controls the evolution of states of network nodes. The dynamics introduce uncertainty about the system environment, which can be addressed by online learning strategies that infer the system behaviour before taking the appropriate adaptation actions or decisions.

We consider the particular problem of optimal source placement in order to maximmize a reward function on a network, which depends on network processes that are a priori unknown and must be learned online. We address this challenge by blending together online learning theory [18] and Graph Signal Processing (GSP) with the key intuition that the latter permits to appropriately model the large-scale network processes via sparse graph spectral representations. This generates a data-efficient learning framework, whose learning rate does not scale with the dimension of the network as in most methods of the literature, but rather with the dimension of the spectral representation. Indeed, in classical online learning solutions such as those casted as Multi-arm bandit (MAB) problems, the main learning steps (i.e., observation, model refinement and action selection) happen in the action (or node) domain and do not scale properly with the search space. The key intuition underpinning our new framework is to consider these learning steps at the crossroad of the search space (or node) domain and the latent space (or spectral) domain. An agent takes sequential decision strategies in the high-dimensional vertex domain based on the uncertainty of the model estimated in the low-dimensional spectral domain. More specifically, we model the search space as a graph and the action represents the activation of specific nodes on the graph. The resulting reward is mapped as a resultant signal on the graph, which can be sparsely represented in the graph spectral domain. We can thus exploit tools from GSP to sparsely represent high-dimensional signals (rewards or actions) as a combination of graph basis functions. With respect to classical MAB problems, we provide a novel theoretical framework that cast sparse graph signal representations into classical linear MAB problems, where one has to learn a graph spectral model online. As a result, the learning process boils down to inferring spectral graph representations with a learning rate that scales with the dimension of their generating kernels, which is substantially lower than the one of the search space.

This new general online learning framework is key in solving our source optimization problem. We consider settings where a central agent places advertisements on a few strategic users within a large social networks, in such a way that the spreading/sharing of the ads across the network is maximized. The vertex domain represents in this case the network of users, with as many nodes as users. Conversely the spectral domain it is rather the latent space domain, which characterizes the ads evolution over the network via graph spectral filtering. With our framework, this online learning problem can be reformulated and reduced to a linearUCB problem [10] {**PF:** *+short characteristics/benefits in 2 words*}. We then derive the theoretical bound of the estimation of the graph spectral model and translate it to the MAB upper confidence bound. Finally, we observe that the optimization method leads to an arm selection problem that is NP-hard, and we provide a low-complexity algorithm by exploiting the structure of the optimization function (maximization of a convex function over a polytope). Simulation results validate the accuracy of the proposed low-complexity algorithm as well as the gains of the proposed graph-kernel MAB strategy, in terms of cumulative regret, sample efficiency and computational complexity, when compared to baseline offline methods that typically separate the learning and testing phases.

The reminder of this paper is as follows {**PF:** *to complete once the whole text is finalized.*}

{**PF:** *clarify name for the proposed algorithm*}

2 Graph-Kernel MAB Framework

Let consider a learner (or agent) controlling processes on large scale networks with no *a priori* information on their dynamics. Examples can be network cooling systems [15], opinions spreading across social networks [28], or energy distribution networks [29] that need to be managed online with no a priori information about the underlying processes. In this paper, we model these processes as signals on graphs. Namely, we denote by $\boldsymbol{w}_t$ the instantaneous reward of $\boldsymbol{a}_t$, the action taken by the learner at time t. Assuming that the action is modelled as an excitation signal on the graph, the reward $\boldsymbol{w}_t = \mathcal{V} \to \mathbb{R}^N$ can modelled as a function of a graph signal resultant from action taken at t. In Fig. 13 box "**Graph-Kernel MAB problems**", with actions and resultant signal defined on the weighted and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ with $\mathcal{V}$ being the vertex set ($|\mathcal{V}| = N$), $\mathcal{E}$ the edge sets, and W the $N \times N$ graph adjacency matrix.

As pointed out in the previous section, learning the mapping $\boldsymbol{a}_t \to r(\boldsymbol{r}_t)$ with current MAB-based approaches would lead to sublinear regret scaling with the search or action space. This is not a feasible solution in large-scale systems. In this work, we propose a graph-kernel MAB problem that exploits the geometry of the network processes to achieve a better regret scaling. Specifically, we write $r(\boldsymbol{r}_t) = \boldsymbol{w}_t$ and model the mapping $\boldsymbol{a}_t \to \boldsymbol{w}_t$ as an *unknown* structured function of the graph Laplacian $\boldsymbol{L}$, i.e., $\boldsymbol{w}_t = f\left(g_L(\boldsymbol{a}_t)\right) + \boldsymbol{n}_t$, with $g_L(\cdot)$ being an *unknown* generating kernel[3] of the graph Laplacian $\boldsymbol{L}$ and $f(\cdot)$ being an affine function[4]. The generating kernel models the process on graphs and characterizes the effect of an action in a resulting graph signal, which will impact the mean reward. Hence, the agent infers the mapping $\boldsymbol{a}_t \to r\left(\boldsymbol{a}_t\right)$ by learning the graph generating

[3] Graph filter defined in the spectral domain of the graph, typically in the form of the power series of the graph Laplacian {**PF:** *+ref?*}.

[4] This include many reward shapes such as subsampled or filtered signal as well as mean value.

kernel $g_L(\cdot)$ in the spectral domain, which is much more sample-efficient than learning the mapping $\boldsymbol{a}_t \rightarrow r(\boldsymbol{a}_t)$ directly in the high-dimensional vertex (action) space.

We now formulate the online learning problem via graph signal processing tools, and we cast the problem into a linear MAB problem, in which the confidence bound is defined on the graph spectral parameters of the generating kernel. We model the network process via the graph-based parametric dictionary learning algorithm in [35], with a signal on graph defined as $\boldsymbol{y} = \boldsymbol{\mathcal{D}}\boldsymbol{h} + \boldsymbol{\epsilon}$, with $\boldsymbol{h} = [h_1, h_2, \dots, h_N]^T$ being the latent variables (localized events) defined on the graph, i.e., the excitation signal defined as actions in our model. The graph dictionary $\boldsymbol{\mathcal{D}}$ is defined as $\boldsymbol{\mathcal{D}} = g_{\boldsymbol{L}}(\cdot) = \sum_{k=0}^{K-1} \alpha_k \boldsymbol{L}^k$ [35] and represents the graph kernel, which incorporates the intrinsic geometric structure of data domain into the atoms of the dictionary through $\boldsymbol{L}$. Assuming that signals have a support contained within K hops from vertex n, the resulting signal at vertex n can be represented as combinations of localized events (e.g., local signals) on the graph, which can appear in different vertices and diffuse along the graph. Namely,

$$y_n = \sum_{m=1}^{N} h_m \sum_{k=0}^{K-1} \alpha_k (L^k)_{n,m} + \epsilon_n \tag{1}$$

where $(\boldsymbol{L}^k)_{n,m}$ is the (m,n) entry of $\boldsymbol{L}^k$ and we recall that $(\boldsymbol{L}^k)_{n,m} = 0$ if the shortest path between n and m has a number of hops that is greater than k. Finally, $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \dots, \epsilon_N]^T$ is a Gaussian and N-dimensional random variable with $\epsilon_n \sim \mathcal{N}(0, \sigma_e^2)$ [10]. With the following matrix notations where $\boldsymbol{P} = [\boldsymbol{L}^0, \boldsymbol{L}^1, \boldsymbol{L}^2, \dots, \boldsymbol{L}^{K-1}]$, with $\boldsymbol{P} \in \mathbb{R}^{N \times NK}$, captures the powers of the Laplacian, and with $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, \dots, \alpha_{K-1}]^T$ representing the polynomials coefficients in the dictionary, we can rewrite the resulting signal as

$$\boldsymbol{y} = g_L(\boldsymbol{h}; \boldsymbol{\alpha}) = \boldsymbol{P}\boldsymbol{I}_K \otimes \boldsymbol{h}\boldsymbol{\alpha} + \boldsymbol{\epsilon} = \boldsymbol{P}\boldsymbol{H}\boldsymbol{\alpha} + \boldsymbol{\epsilon} \tag{2}$$

with $\boldsymbol{I}_K$ being the $K \times K$ identity matrix, $\otimes$ the Kronecker product, and $\boldsymbol{H} = \boldsymbol{I}_K \otimes \boldsymbol{h}$, with $\boldsymbol{H} \in \mathbb{R}^{NK \times K}$. Without loss of generality, in the following we assume that the agent controls the latent variables $\boldsymbol{h}$ while learning the polynomial coefficients $\boldsymbol{\alpha}$. However, the problem formulation could be also extended to the reverse case in which gent controls the dynamics of the network process via $\boldsymbol{\alpha}$ while learning the input latent variables $\boldsymbol{h}$. Given that the instantaneous reward is a linear function of the resultant signal $\boldsymbol{y}$, substituting (2) in the reward expression we achieve the following

$$\boldsymbol{w} = \boldsymbol{M}\boldsymbol{y} = \boldsymbol{M}\boldsymbol{P}\boldsymbol{H}\boldsymbol{\alpha} + \boldsymbol{M}\boldsymbol{\epsilon} = \boldsymbol{X}\boldsymbol{\alpha} + \boldsymbol{n} \tag{3}$$

where and $\boldsymbol{X} = \boldsymbol{M}\boldsymbol{P}\boldsymbol{H}$, with $\boldsymbol{X} \in \mathbb{R}^{N \times K}$. In short, the reward can be expressed as a linear combination of the K-degree polynomial $\boldsymbol{\alpha}$ and the matrix $\boldsymbol{X}$, which includes both the graph structure information (via the Laplacian $\boldsymbol{L}$) and the action $\boldsymbol{h}$. The above reward is important for two key reasons:

– it has a linear mapping similar to (31), implying that we can solve the online learning problem with the linUCB theory, (32), where the confidence ellipsoid is however defined in the graph spectral domain.
– the reward is given by the generating kernel $g_L(\cdot)$, which is parametrized by the vector $\boldsymbol{\alpha}$ with dimensionality K. It follows that the agent needs to learn a K (low) dimensional polynomial instead of learning a high-dimensional mapping.

3 Online Source Optimisation Problem

We now propose a theoretical bound and algorithmic solution to the online source optimisation problem using the new framework described in the previous section. Specifically, we consider the problem where a decision maker needs to select T_0 sparse actions out of N, i.e., $\mathcal{A} = \{\boldsymbol{h} \mid ||\boldsymbol{h}||_0 \leq T_0 \ \wedge \ h_n \in [0,1], \ n = 1, ..., N\}$, where T_0 is the maximum sparsity level of the actions. Without loss of generality, we assume that the rewards associated to consecutive actions are i.i.d.. We then define the mapping function $\boldsymbol{M} \in \mathbb{R}^{N \times N}$ as a diagonal binary matrix, with the n-th diagonal element being 1 if the signal at the node n is observed, 0 if the signal is masked. Applicative examples are influence maximization problems, such as placing ads to maximize the product appreciation over time with a network of users (e.g., social network) [34]. Reward would be the users feedback or click-on-ads (resultant signal), which realistically is observed only for some users ($\boldsymbol{M}$ being a masking matrix). Another example is the optimization of cooling systems and/or power networks, in which energy sources need to be optimized favoring global network heating system. This online source optimization on large-scale networks can be formulated as $\max_{\{\boldsymbol{h}_t\}_t} \sum_{t=1}^{T} r(\boldsymbol{h}_t)$ subject to the action belonging to the action set $\mathcal{A}$.

The problem can be casted as a stochastic MAB problem, aimed at minimizing the cumulative loss (or equivalently maximize the cumulative reward), which is seen as the minimization of the pseudo regret $R_T = Tr(\boldsymbol{h}^{\star}) - \sum_{t=1}^{T} r(\boldsymbol{h}_t)$. Classical MAB problems achieve a sublinear regret, i.e., $R_T = \mathcal{O}(|\mathcal{A}| \log T)$ [18], with $|\mathcal{A}| = \binom{N}{T_o}$, if T_0 is the imposed sparsity of $\boldsymbol{h}$, Fig. **??**. This regret is not sustainable in large-scale networks (i.e., with large action space $|\mathcal{A}|$). In the following we formalize the problem in our new framework of Section 2, by placing sources and observing the resulting signal in the network domain while learning the model $\boldsymbol{\mathcal{D}}$ in a sparse graph kernel domain, see Fig. **??**.

We now propose a novel algorithm for source optimization using the framework of Section III, which permits to learn in the spectral domain and eventually to act 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.

Step 1: Coefficients estimation Let consider the t-th decision opportunity, when $t-1$ decisions have already been taken and the corresponding signals and rewards have been observed. The training set built over time thus corresponds to sequence of pairs $\{(\boldsymbol{h}_\tau, \boldsymbol{w}_\tau)\}_{\tau=1}^{t-1}$, where we recall that $p(\boldsymbol{y}|\boldsymbol{h},\boldsymbol{\alpha}) \sim \mathcal{N}(g_L(\boldsymbol{h};\boldsymbol{\alpha}), \sigma_e^2 \boldsymbol{I}_N)$, where the randomness is due to the random noise $\boldsymbol{\epsilon}_\tau$. For large t, maximizing the MAP probability $p(\boldsymbol{\alpha}|\boldsymbol{y},\boldsymbol{h})$ corresponds to minimize the l^2-regularized least-square estimate of $\boldsymbol{\alpha}$, which leads to the following estimation problem:

$$\hat{\boldsymbol{\alpha}}_t : \arg\min_{\boldsymbol{\alpha}} \sum_{\tau=1}^{t-1} ||\boldsymbol{MPH_\tau \alpha} - \boldsymbol{w}_\tau||_2^2 + \lambda ||\boldsymbol{\alpha}||_2^2 \,. \tag{4}$$

It follows that

$$\hat{\boldsymbol{\alpha}}_t = \left[\sum_{\tau=1}^{t-1} \boldsymbol{Z}_\tau^T \boldsymbol{Z}_\tau + \lambda \boldsymbol{I}_K\right]^{-1} \sum_{\tau=1}^{t-1} \boldsymbol{Z}_\tau^T \boldsymbol{w}_\tau = \left[\boldsymbol{Z}_{1:t}^T \boldsymbol{Z}_{1:t} + \lambda \boldsymbol{I}_K\right]^{-1} \boldsymbol{Z}_{1:t}^T \boldsymbol{W}_t = \boldsymbol{V}_t^{-1} \boldsymbol{Z}_{1:t}^T \boldsymbol{W}_t \tag{5}$$

with $\boldsymbol{Z}_{1:t} = [\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_{t-1}]^T$, $\boldsymbol{Z}_\tau = \boldsymbol{MPH}_\tau$, $\boldsymbol{W}_t = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{t-1}]^T$, and $\boldsymbol{V}_t = \boldsymbol{Z}_{1:t}^T \boldsymbol{Z}_{1:t} + \lambda \boldsymbol{I}_K$. 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 (4) 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 best action to take for the t-th decision opportunity. Following the theory of linear UCB [10], the decision maker evaluates the confidence bound E_t as an ellipsoid centered in $\hat{\boldsymbol{\alpha}}_t$ defined such that $\boldsymbol{\alpha} \in E_t$ with probability $1-\delta$ for all $t \geq 1$, see Fig. 14. Then, the decision maker will select the best action that maximizes the estimated mean reward, for each possible generating kernel in the ellipsoid. Formally, the decision maker selects the action $\boldsymbol{h}$ (and therefore $\boldsymbol{X} = \boldsymbol{MPI}_K \otimes \boldsymbol{h}$) such that

$$\boldsymbol{h}_t : \arg\max_{\boldsymbol{h}\in\mathcal{A}} \max_{\boldsymbol{\alpha}\in E_t} \boldsymbol{X\alpha} \,. \tag{6}$$

The intuition is that rather than maximizing the reward $r(\boldsymbol{h}) = \boldsymbol{MPI}_{K+1} \otimes \boldsymbol{h}\hat{\boldsymbol{\alpha}}_t = \boldsymbol{X}\hat{\boldsymbol{\alpha}}_t$, the decision maker takes into account the uncertainty on the estimate of the polynomial and looks at all possible possible generating kernels in E_t (optimism in face of uncertainty [18]). However, to apply (6), we need to formally derive the confidence bound E_t. This can be derived by the following two Lemmas (proofs in Appendix 9).

Lemma 1 bounds the matrix $\boldsymbol{V}_t$, which defines the regularized least-square solution as shown in (5). Lemma 1 is key to evaluate the upper confidence bound in Lemma 2. Specifically, Lemma 2 provides the confidence bound E_t such that $E_t : \{||\hat{\boldsymbol{\alpha}}_t - \boldsymbol{\alpha}_*|| \leq c_t\}$. It is worth noting that both bounds have explicit dependency on topological features of the graph, such as the sum of eigenvalues power, as we comment later.

Lemma 1: *Suppose* $\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_t \in \mathbb{R}^{1\times K}$*, with* $\boldsymbol{Z}_\tau = \boldsymbol{MPI}_K \otimes \boldsymbol{h}_\tau$ *and for any* $1 \leq \tau \leq t-1$*,* $||h_\tau||_F^2 \leq T_0$*, and* $||\boldsymbol{M}||_F^2 \leq Q$*. Let* $\boldsymbol{V}_t = \sum_\tau \boldsymbol{Z}_\tau^T \boldsymbol{Z}_\tau + \lambda \boldsymbol{I}_K$

Algorithm 1 Kernel-UCB

```
Input:
$N$: nr of nodes, $T_0$: sparsity level of initial signal $\boldsymbol{h}$, $K$: sparsity of the basis coefficients
$\lambda, \delta$: 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
    $\boldsymbol{X}_{1:t} = [\boldsymbol{X}_1, \boldsymbol{X}_2, \dots, \boldsymbol{X}_{t-1}]^T$
    $\boldsymbol{Y}_{1:t} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{t-1}]^T$
    $\boldsymbol{V}_t = \boldsymbol{X}_{1:t}^T \boldsymbol{X}_{1:t} + \lambda \boldsymbol{I}_{K+1}$
    $\hat{\boldsymbol{\alpha}}_t = \boldsymbol{V}_t^{-1} \boldsymbol{X}_{1:t}^T \boldsymbol{Y}_{1:t}$
    Evaluate the confidence bound and select the best action
    $c_t = R\left[\sqrt{K \log(1 + tNT_0 d/\lambda)} + \sqrt{2 \log 1/\delta}\right] + \lambda^{1/2} S$
    $\boldsymbol{h}_t$ : $\arg\max_{\boldsymbol{h} \in \mathcal{A}} \left[\boldsymbol{MPH}\hat{\boldsymbol{\alpha}}_t + c_t ||\boldsymbol{MPH}||_{\boldsymbol{V}_t^{-1}}\right]$
    Observe the resulting signal $\boldsymbol{y}_t$ and the instantaneous reward $r(\boldsymbol{y}_t)$
    $t = t + 1$
end while
```

with $\lambda > 0$, then $|\boldsymbol{V}_t| \leq [\lambda + tdQT_0]^K$, with $d = \sum_k \sum_l \lambda_l^k$, with λ_l being the l-th eigenvalue of the graph Laplacian.

Lemma 2: *Assume that $\boldsymbol{V}_t = \sum_\tau \boldsymbol{Z}_\tau^T \boldsymbol{Z}_\tau + \lambda \boldsymbol{I}_K$, define $\boldsymbol{w}_\tau = \boldsymbol{Z}_\tau \boldsymbol{\alpha_*} + \boldsymbol{\eta}_\tau$, with $\boldsymbol{Z}_\tau = \boldsymbol{MPI}_K \otimes \boldsymbol{h}_\tau$ and with $\boldsymbol{\eta}_t$ being conditionally R-sub-Gaussian, and assume that $||\boldsymbol{\alpha}_*||_2 \leq S$, and $||\boldsymbol{h}_\tau||_F^2 \leq T_0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, for all $t \leq 0$, $\boldsymbol{\alpha}_*$ lies in the set*

$$E_t : \left\{\boldsymbol{\alpha} \in \mathbb{R}^{1\times K} : ||\hat{\boldsymbol{\alpha}}_t - \boldsymbol{\alpha}||_{\boldsymbol{V}_t} \leq R\left[\sqrt{K \log(\lambda + tdQT_0) + 2\log(\lambda^{-1/2}\delta)}\right] + \lambda^{1/2} S\right\}$$

with $d = \sum_k \sum_l \lambda_l^k$, with λ_l being the l-th eigenvalue of the graph Laplacian and $\hat{\boldsymbol{\alpha}}_t$ is the l^2-regularized least-square estimate of $\boldsymbol{\alpha}$ when t training samples are available.

Defining the confidence bound E_t such that $E_t : \{||\hat{\boldsymbol{\alpha}}_t - \boldsymbol{\alpha}_*|| \leq c_t\}$, the maximization in (6) becomes (see Appendix 12 for details)

$$\begin{aligned}
\boldsymbol{h}_t &= \arg\max_{\boldsymbol{h} \in \mathcal{A}} \max_{\alpha \in E_t} \boldsymbol{X\alpha} \\
&= \arg\max_{\boldsymbol{h} \in \mathcal{A}} \boldsymbol{X\alpha} + c_t \sqrt{\boldsymbol{X V}_t^{-1} \boldsymbol{X}^T} \\
&= \arg\max_{\boldsymbol{h} \in \mathcal{A}} \boldsymbol{X\alpha} + c_t ||\boldsymbol{X}||_{\boldsymbol{V}_t^{-1}} \\
&= \arg\max_{\boldsymbol{h} \in \mathcal{A}} \left[\boldsymbol{MPH}\hat{\boldsymbol{\alpha}}_t + c_t ||\boldsymbol{MPH}||_{\boldsymbol{V}_t^{-1}}\right] \qquad (7)
\end{aligned}$$

with $c_t = R\left[\sqrt{K\log(\lambda + tdQT_0) + 2\log(\lambda^{-1/2}\delta)}\right] + \lambda^{1/2}S$ following Lemma 2. This optimization characterizes the Step 2, *i.e.*, the action selection. In Algorithm 1, we summarize the main steps of the proposed kernel-UCB strategy.

We now derive the regret bound. From [38], we have

$$R_T \leq 2(c_T+1)\sqrt{T\sum_{t=1}^{T}\min\left(1, ||\boldsymbol{X}||^2_{V_t^{-1}}\right)} \tag{8}$$

and from Lemma 11 in [1], we also have

$$\min\left(1, ||\boldsymbol{X}||^2_{V_t^{-1}}\right) \leq 2\log(|\boldsymbol{V}_t|/\lambda \boldsymbol{I}_{K+1}|) \tag{9}$$

Substituting (9) in (8) and applying Lemma 1, we get

$$R_T \leq 2(c_T+1)\sqrt{T\sum_{t=1}^{T}2K\log\left(1+\frac{NT_0 d}{\lambda}\right)}. \tag{10}$$

4 Online Source Optimization Algorithm

The methodology proposed in the previous Section entails two main optimization/learning steps that need to be solved. While the solution to the optimization in Step 1 has a closed form solution, i.e., equation (5), in Step 2 the optimization problem in (7) needs to be solved efficiently. This optimization becomes computationally expensive in large-scale graphs, see Appendix C, therefore in the following we propose a computationally effective optimization algorithm. We first rewrite the problem as follows

$$\begin{aligned} &\max_{\boldsymbol{h}} \boldsymbol{D}\boldsymbol{h} + c_t\sqrt{\boldsymbol{X}\boldsymbol{V}_t^{-1}\boldsymbol{X}^T} \\ &\text{s.t. } h(n) \in [0,1], \ \forall n \\ &\quad ||\boldsymbol{h}||_0 \leq T_0 \end{aligned} \tag{11}$$

where the constraints for $\boldsymbol{h}$ are explicitely related to $\mathcal{A}$, and where $\boldsymbol{D}\boldsymbol{h} = \boldsymbol{X}\boldsymbol{\alpha}$, according the notation introduced in Sec. 2. Decomposing $\boldsymbol{V}_t^{-1}$ as $\boldsymbol{V}_t^{-1} = \boldsymbol{L}^T\boldsymbol{L}$, we have $\boldsymbol{X}\boldsymbol{V}_t^{-1}\boldsymbol{X}^T = ||\boldsymbol{L}\boldsymbol{X}^T||_2^2$. This leads to the following optimization problem

$$\begin{aligned} &\max_{\boldsymbol{h}} \boldsymbol{D}\boldsymbol{h} + c_t||\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T||_2 \\ &\text{s.t. } h(n) \in [0,1], \ \forall n \\ &\quad ||\boldsymbol{h}||_0 \leq T_0 \end{aligned} \tag{12}$$

where we have used $\boldsymbol{X} = \boldsymbol{M}\boldsymbol{P}\boldsymbol{I}_{K+1} \otimes \boldsymbol{h} = \boldsymbol{b} \otimes \boldsymbol{h}$. The above problem maximizes a convex objective function over a polytope, defined by both 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.

However, in the case of large networks, this computation could be too expensive. Therefore, we propose an algorithm that walks along the edges of the polytope. The intuition is similar to the one of the simplex algorithm or any hill climbing algorithm. Let consider an iterative algorithm, where at each iteration the N variables h_n, with $n = 1, \dots, N$, are subdivided into basic variables and non-basic variables. The former are the ones such that $h_n = 1$, while the non basic variables are the remaining zero sources. At each iteration we perform the operation of moving from a feasible solution to an adjacent feasible solution by swapping a basic variable with a non basic one (similar to the pivoting operation in the simplex algorithm). We move in such a way that the objective function always increases. We then stop the algorithm either after a maximum number of iteration steps or when convergence has reached.

We now describe the algorithm in more details. Let define

$$\boldsymbol{h}^{(t-1)} = [h_1^{(t-1)}, h_2^{(t-1)}, \dots, h_N^{(t-1)}]$$

be the optimal variable at the iteration step $i-1$. Let $\mathcal{B}_t = \{n | h_n^{(t-1)} = 1\}$ be the set of the indices of basic variables at t. Let then denote by J the objective function $J(\boldsymbol{h}) = \boldsymbol{MPI}_{K+1} \otimes \boldsymbol{h}\hat{\boldsymbol{\alpha}}_t + c_t ||\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T||_2$ and by $\partial J / \partial h_n$ be the partial derivative of J with respect the nth variable. Finally, note that vertices are adjacent if they share all but one non-basic variable. Equipped with the above notations and definitions, we now state the following Lemmas (further explained in Appendix 10).

Lemma 3: *One vertex is optimal if there is no better neighboring vertex.*

Lemma 4: *From a vertex, moving to one of the neighboring nodes in the direction of the greatest gradient leads to a no-worse objective function. Let $h_{in}^{(t)}$ and $h_{out}^{(t)}$ be the variable that enters and leaves the set of basic variables, respectively, at the t-th iteration. These variables are evaluated as follows*

$$in = \arg \max_{n | h_n \notin \mathcal{B}_t} \left\{ \frac{\partial J}{\partial h_n}\Big|_{\boldsymbol{h}^{(t-1)}} \right\}, \quad out = \arg \min_{n | h_n \in \mathcal{B}_t} \left\{ \frac{\partial J}{\partial h_n}\Big|_{\boldsymbol{h}^{(t-1)}} \right\}$$

Therefore, given a vertex $\boldsymbol{h}^{(t-1)}$, at the t-th iteration the algorithm will move to the neighboring vertex $\boldsymbol{h}^{(t)}$ defined as follows:

$$h_{\text{in}}^{(t)} = 1, \quad h_{\text{out}}^{(t)} = 0, \quad \text{and} \quad h_n^{(t)} = h_n^{(t-1)}, \forall n \neq in, out.$$

As shown in Algorithm 2 that solves (12), if the swap variable leads to an improvement of the objective function, *i.e.*, if $J(\boldsymbol{h}^{(t)}) > J(\boldsymbol{h}^{(t-1)})$, then we proceed to the next step. Otherwise, we set the optimal source signal $\boldsymbol{h}^\star = \boldsymbol{h}^{(t-1)}$ and we break the iterative loop. Further details are provided in Algorithm 2, together with the initialization step. Rather than a randomly generating starting point, we consider the one with the T_0 variables having the maximum partial derivative

Algorithm 2 Algorithm for Action Selection

```
Input:
number of iterations $MaxIter$, sparsity level $T_0$, graph topology (and therefore $\boldsymbol{L}$
and $\boldsymbol{P}$), reward mask $\boldsymbol{M}$, estimated polynomial $\hat{\boldsymbol{\alpha}}$, confidence bound $c$.
Output:
optimal source signal $\boldsymbol{h}^\star$
Initialization:
Definition of the objective function $J(\boldsymbol{h}) = \boldsymbol{MPI}_{K+1} \otimes \boldsymbol{h}\hat{\boldsymbol{\alpha}} + c||\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T||_2$
Evaluation of the partial derivatives $a_n = \frac{\partial J}{\partial h_n}\big|_{\boldsymbol{u_n}}, \forall n$
Selection of $\boldsymbol{h}^{(0)}$: $h_n^{(0)} = 1$ if $a_n$ belongs to the $T_0$ largest partial derivatives.
$t = 1$
for $t \leq MaxIter$ do
    Set $\mathcal{B}_t = \{n | h_n^{(t-1)} = 1\}$
    Evaluate the IN and OUT variables:
        $in = \arg \max_{n|h_n \notin \mathcal{B}_t} \left\{ \frac{\partial J}{\partial h_n}\Big|_{\boldsymbol{h}^{(t-1)}} \right\}, \quad out = \arg \min_{n|h_n \in \mathcal{B}_t} \left\{ \frac{\partial J}{\partial h_n}\Big|_{\boldsymbol{h}^{(t-1)}} \right\}$
    Set $\boldsymbol{h}^{(t)} = \boldsymbol{h}^{(t-1)}$
    Set $h_{in}^{(t)} = 1, h_{out}^{(t)} = 0$
    if $J(\boldsymbol{h}^{(t)}) \leq J(\boldsymbol{h}^{(t-1)})$ then
        $\boldsymbol{h}^\star = \boldsymbol{h}^{(t)}$
        break
    end if
    $t \leftarrow t + 1$
end for
```

$a_n = \frac{\partial J}{\partial h_n}\big|_{\boldsymbol{u_n}}, \forall n$, with $\boldsymbol{u_n}$ being a N-dimensional vector all elements null but the n-th, which is set to 1.

The remaining step is the evaluation of the partial derivatives. We recall that the objective function is given by

$$J(\boldsymbol{h}) = \underbrace{\boldsymbol{MPI}_{K+1} \otimes \boldsymbol{h}\hat{\boldsymbol{\alpha}}}_{F(\boldsymbol{h})} + c \underbrace{||\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T||_2}_{G(\boldsymbol{h})} .$$

We now evaluate the partial derivative for each of the two components of the objective function. Namely:

$$\begin{aligned} \frac{\partial F(\boldsymbol{h})}{\partial h_n} &= \frac{\partial}{\partial h_n} \left(\boldsymbol{MPI}_{K+1} \otimes \boldsymbol{h}\hat{\boldsymbol{\alpha}} \right) \\ &= \boldsymbol{MP} \frac{\partial}{\partial h_n} \left(\boldsymbol{I}_{K+1} \otimes \boldsymbol{h} \right) \hat{\boldsymbol{\alpha}} \\ &= \boldsymbol{MPI}_{K+1} \otimes \boldsymbol{1}_n \hat{\boldsymbol{\alpha}} \end{aligned} \tag{13}$$

$$\begin{aligned}\frac{\partial G(\boldsymbol{h})}{\partial h_n} &= ||\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T||_2 \\ &= \frac{\left(\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T\right)}{||\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T||_2} \frac{\partial \left(\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T\right)}{\partial h_n} \\ &= \frac{\left(\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T\right)}{||\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T||_2} \left(\boldsymbol{L} * \boldsymbol{b}^T \otimes \mathbf{1}_n^T\right)^T \end{aligned} \quad (14)$$

This leads to the following partial derivative

$$\frac{\partial J(\boldsymbol{h})}{\partial h_n} = \boldsymbol{MPI}_{K+1} \otimes \mathbf{1}_n \hat{\boldsymbol{\alpha}} + \frac{\left(\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T\right)}{||\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T||_2} \left(\boldsymbol{L} * \boldsymbol{b}^T \otimes \mathbf{1}_n^T\right)^T \quad (15)$$

In summary, the proposed algorithm requires the evaluation of the partial derivative (N operations) instead of exhaustively evaluating the objective function in (12) at all $\binom{N}{T_0}$ possible edges. In the following section, we empirically validate the performance of the above algorithm. {**PF:** *should we give a name to the proposed algorithm, wrap up on a short description (summary of integration of above steps in 2 words), and make connections with results later - maybe be helpful to clarify - see comments later in Results section.*}

5 Simulation Results

5.1 Settings

We now evaluate experimentally our proposed online source optimization algorithm that is based on the new Graph-Kernel MAB framework. As benchmark solution, we propose an algorithm denoted as Act After Learning (AAL), in which the exploration and the exploitation phases are separated, while our proposed method finds the best tradeoff between exploitation and exploration automatically. The key intuition is that it first gathers a training set (in the first T_L decision strategies) and therefore experience a reward as a function of random actions. Then, after a training phase of T_L decision opportunities, the generating kernel is estimated and the best arm is selected. In the remaining decision opportunities the best action is taken and the reward observed.

We carry out experiments on Barabási-Albert model (BA) graphs [4], on radial basis function (RBF) random graphs, and on non-synthetic graphs (*e.g.*, Minnesota graph[5]). For BA graphs, the network begins with an initial connected network of $m_0 = 10$ nodes. At each iteration, one node is added to the network and it is connected to $m \leq m_0$ existing nodes. Connections to existing nodes are built following a preferential attachment mechanism, which eventually builds a scale-free network. For the RBF model, 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

[5] Available at `https://lts2.epfl.ch/gsp/`.

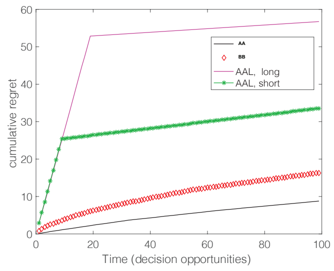
smaller than or equal to T, and zero otherwise. We further set $\sigma = 0.5$ and we vary T to change the edge density of the generated graphs.

If not specified otherwise, we consider that network processes take the form of diffusion processes on the graphs described above. We then consider that each signal on the graph is characterized by the source signal, the generating kernel and an additive random noise ϵ_t with zero mean and variance σ_e^2 (i.e., $R = \sigma_e$ in the spectral UCB). {**PF:** *maybe we can add the formulation of the graph signal in the case of diffusion processes?*} The remaining parameters of the sequential decision strategy are set as $\lambda = 0.01$, $\delta = 0.01$. {**PF:** *should we talk about M too?*}

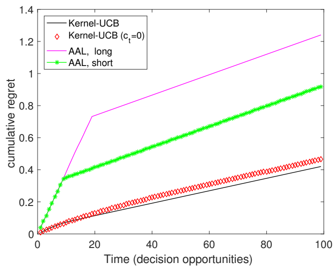
5.2 Performance of Kernel-UCB

We now show the performance of our online learning solution with respect to the AAL baseline algorithm. We study the performance of the proposed Kernel-UCB, assuming that the optimization in Step 2 is solved by Algorithm 2. First, a randomly generated graph (RBF model) with $N = 100$ nodes is considered, in the case of diffusion process acting on the graph with $\tau = 10$ and with $\sigma_e^2 = 10^{-2}$. Fig. 1 depicts the cumulative regret over 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 are generated). The proposed Kernel-UCB is provided for both the case with and without confidence bound, *i.e.*, c_t as evaluated from Algorithm 1, or $c_t = 0$. Then, the Kernel-UCB is compared to the baseline algorithm 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 $\boldsymbol{\alpha}$. But also the longer the suboptimal phase, since during the learning phase actions are selected at random. From Fig. 1, we observe that the Kernel-UCB (both with and without confidence bound) outperforms the baseline algorithms. We also notice that the baseline with $T_L = 10$ leads to an estimate of the polynomial $\boldsymbol{\alpha}$ that is not accurate enough. Therefore, after the learning phase the decision-maker selects future actions with a wrong estimate of $\boldsymbol{\alpha}$. This leads to a large level of suboptimality and therefore to a rapidly increasing cumulative regret.

In Fig. 2, the cumulative regret is provided as a function of time for the same settings of Fig. 1, but with $\tau = 10$ and $\tau = 0.5$ for the diffusion process. Larger τ means a more diffused (and therefore more informative) resulting signal. Therefore, in the case of $\tau = 10$ the estimate of the polynomial $\boldsymbol{\alpha}$ is less challenging than the case of diffusion process with $\tau = 0.5$. This can be observed by two behaviors: 1) a learning time of 20 decision opportunities does not improve the estimation with respect to a learning framework of 10 decision opportunities. This is motivated by the fact that the two AAL algorithms have the same slope of the cumulative regret. 2) The cumulative regret evaluated with the Kernel-UCB with $c_t = 0$ almost overlaps with the curve obtained from Kernel-UCB with confidence bound. This means that taking into account the uncertainty of the estimation is less beneficial in Fig. 2(a), and this is due to the better estimate of the polynomial. On the contrary, in the case of a more localized process, namely

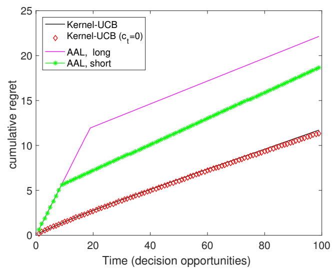


(a) Random Graph, Polynomial Dictionary ($K = 20$)

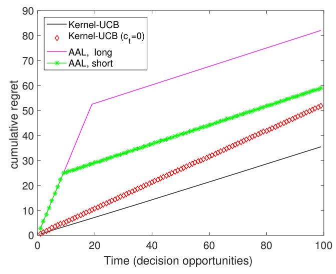


(b) Community Graph, Diffusion dictionary, $\tau = 5$.

Fig. 1. Cumulative regret vs. time for randomly generated graphs with $N = 100$, diffusion process (with $\tau = 5$) and sparsity level $T_0 = 5$.



(a) $\tau = 10$

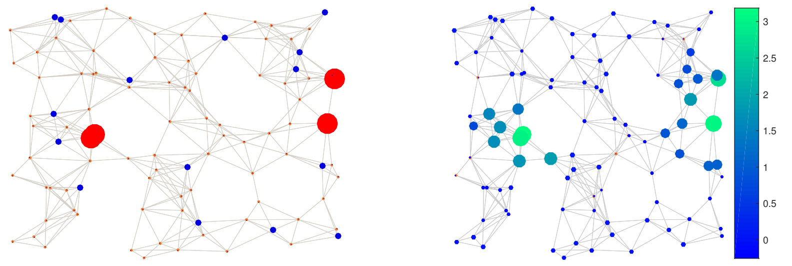


(b) $\tau = 0.5$

Fig. 2. Cumulative regret vs. time for randomly generated graphs with $N = 100$, diffusion process and sparsity level $T_0 = 5$.

$\tau = 0.5$, the estimation process is more challenging, hence taking into account the confidence bound leads to a better selection of the actions $\boldsymbol{h}$ over time.

We further illustrate in Fig. 3 that optimal placement of sparse resources in high dimensional networks is not necessarily an intuitive step. It depicts the optimal source signal computed by our algorithm {**PF:** *it is computed by oure algorithm, right?*} and the resulting signal for a randomly generated graph (RBF model) with $N = 100$, and sparsity level $T_0 = 4$. In Fig. 3(a), the optimal signal is depicted in red, while the mask signal used to evaluate the reward is depicted in blue. The mask M is randomly generated and it covers 20% of the nodes. It is worth noting that the optimal signal is placed on nodes that do not necessarily belong to the mask and do not necessarily appear to be central in the graph. However, this results in the optimal reward signal depicted in Fig, 3(b).



(a) Source signal (red) and mask signal M (blue)

(b) Resulting Signal

Fig. 3. Optimal source signal and resulting signal for a randomly generated graph (RBF model) with $N = 100$, and sparsity level $T_0 = 4$.

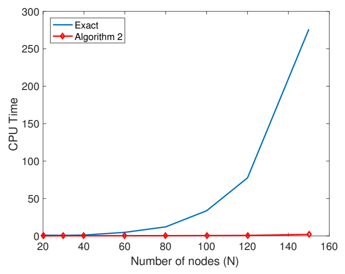
5.3 Empirical Reward-Complexity Tradeoff

We validate now the proposed algorithm for solving the action selection Step, which is a priori NP-hard. We empirically compare the numerical solver (FMICON) adopted to optimally solve Step 2 and the proposed Algorithm 2 {**PF:** *maybe clear is we give a name to each version of the algorithm, and then use it in results discussion - possibly figures - hopefully without need to change the labels.*}. Fig. 4(a) depicts the CPU time required by both solvers as a function of the number of nodes N for a randomly generated graph (RBF model), with sparsity level $T_0 = 5$. The achieved reward after 100 decision steps is also depicted in Fig. 4(b). Results are averaged over 50 generated graphs. As expected, the problem in (28) is NP-hard (maximization of a convex function under convex -or affine- constraints) and the solver's complexity grows exponentially with N. Conversely, the complexity of Algorithm 2 grows linearly with N, as shown by Fig. 4(a). From Fig. 4(b), it can be observed that the proposed solution still achieves the optimal solution in terms of reward. Note that the reward is not monotonic with N because the density of the graph is not necessarily kept constant for different N values. Similar conclusions can be observed from Fig. 5, where both CPU time and reward are provided as a function of the edge density of random graphs with 80 nodes. Results are averaged over 50 generated graphs.

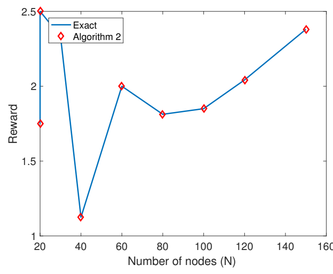
6 Related Work

{**PF:** *to be consolidated, possibly also adding pieces from the Bandits Overview section in Appendix.*}

In decision making strategies (DMS), an agent optimises sequential actions in a way that maximizes the expected reward, when each action's and reward's properties are uncertain a priori. This uncertainty can be reduced over time by learning from experience, striking the well known exploration-exploitation tradeoff. When network or user dynamics are unknown, network management

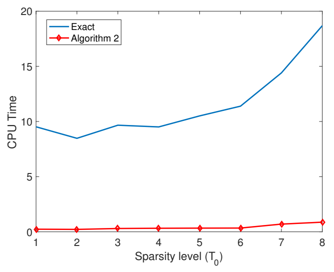


(a) CPU Time

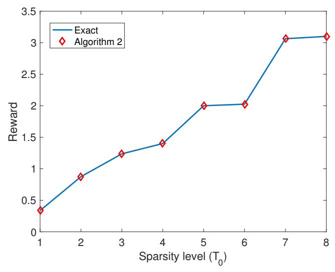


(b) Reward

Fig. 4. Comparison of the optimal solver and the proposed Algorithm 2 for random graphs (RBF model) with different number of nodes, and $T_0 = 5$.



(a) CPU Time



(b) Reward

Fig. 5. Comparison of the optimal solver and the proposed Algorithm 2 for random graphs (RBF model) with $N = 80$ and different sparsity levels.

and optimization problems can be cast as DMS, in which actions represent resource allocation, or source selection strategies [3, 32]. The agent learns a priori unknown stochastic network characteristics while acting, and eventually converges over time to the best action strategy, i.e., the decision that optimizes the network objective function. The key challenge of these networked DMS problems is usually the high dimension of the search and observation spaces. MAB problems are classical DMSs under uncertainty; they are well understood in small-scale environments [9] and they can theoretically be applied to networked bandit settings [11, 17]. However, the regret (action suboptimality) of these problems increases fast with the ambient dimension, which makes the problem intractable in scenarios with infinitely large strategy sets, such as large-scale network optimizations or recommendation systems [18, 42]. Scalability can be addressed by reducing the dimensionality of the search space by clustering arms for example [7, 8, 12, 17, 20–22], but these methods usually perform poorly in irregular datasets [44], typical of most of the real-world problems. This can be addressed by structural learning [5, 24, 33, 37, 41], which identifies and leverages the structure underneath data in optimisation problems where the outputs have semantically rich structure. For example, when MAB problems are applied to online recommendation systems, the social network underpinning users identify users similarity in terms of arms reward. Exploiting this hidden structure improves the efficiency of the learning, however the algorithm still experiences a sublinear regret that scales with the number of network nodes. In the context of decision-making strategies for network processes, it stays an open challenge to design an online learning framework able to capture efficiently the dynamic network processes with a regret that scales properly in high-dimensional regimes.

To the best of our knowledge, this is the first work exploiting GSP tools to handle the irregular network structure of the online learning problems in complex systems. Moreover, no a priori assumption is imposed on the network processes so that our methodology applies to any network processes that can be sparsely represented by graph kernel functions.

This work opens the door to a main stream of research devoted to structured online learning on graphs, where GSP tools are used to learn graph signal models in order to develop new effective decision-making strategies. Note that GSP [27] has been used to capture structural properties of network processes, such as node centrality [14] and community detection [40] for network problems such as diffusion dynamics, pricing experiments, and opinion dynamics. In [40], an unknown network process is modeled as a graph filter that is excited by a set of unknown low-rank inputs/excitations in order to detect communities. Similarly, generative low-pass graph filters have been used for power systems modeling [30]. Despite the growing literature in using GSP tools to infer structural processes, to the best of our knowledge, no work so far has however focused on learning while acting, i.e., inferring network process models while taking sequential actions on those networks.

7 Conclusions

{**PF:** *to consolidate*}

In this work, we study network optimization problems under uncertainty in the case in which one action (or graph signal) leads to a mean reward function of the resulting signal on the graph. The challenge is to optimize sequential actions without knowing a priori the network process that drives the mapping between action and mean reward. The key intuition is to infer the network process form 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 (optimization 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 network online learning problem. As main contributions, we cast the network optimization problem under uncertainty as a linear MAB problem, which infers a K-dimensional polynomial that defines the graph generating-kernel while taking actions over time on the network-graph. We then derive the theoretical bound of the estimation of the graph spectral model and translate it to the MAB upper confidence bound. We show both mathematically and empirically that more connected graphs and sparser signals lead to a more accurate estimation of the network processes. Finally, we observe that the optimization method leads to an arm selection problem that is NP-hard, and we provide a low-complexity algorithm by exploiting the structure of the optimization function. Beyond proposing a data-efficient solution to problems of network optimization, this work aims at opening the gate to new research directions in which graph signal processing tools are blended to online learning frameworks to exploit structural knowledge of network optimization problems.

{**PF:** *Overall, 2 pages should still be cut, maybe by shortening text here and there, consolidating further Sections 2 and 3, or removing a few results (not too much...)*}

8 Ethical Statement

To be completed

Ethics is one of the most important topics to emerge in machine learning and data mining. We ask you to think about the ethical implications of your submission such as, e.g., related to the collection and processing of personal data, the inference of personal information, or the potential use of your work for policing or the military. which will be taken into consideration by the reviewers.As part of your submission, you are asked to include an ethical statement up to one page in length that discusses any ethical implications of your work.

9 Proof of Lemmas

Lemma 1: *Suppose $\boldsymbol{Z}_1, \boldsymbol{Z}_2, \dots, \boldsymbol{Z}_t \in \mathbb{R}^{1\times(K)}$, with $\boldsymbol{Z}_\tau = \boldsymbol{M}\boldsymbol{P}\boldsymbol{I}_K \otimes \boldsymbol{h}_\tau$ and for any $1 \leq \tau \leq t-1$, $||h_\tau||_F^2 \leq T_0$, and $||\boldsymbol{M}||_F^2 \leq Q$. Let $\boldsymbol{V}_t = \sum_\tau \boldsymbol{Z}_\tau^T \boldsymbol{Z}_\tau + \lambda \boldsymbol{I}_K$ with $\lambda > 0$, then $|\boldsymbol{V}_t| \leq [\lambda + tdQT_0]^K$, with $d = \sum_k \sum_l \lambda_l^k$, with λ_l being the l-th eigenvalue of the graph Laplacian.*

Proof: As shown in [1] the following inequality holds $|\boldsymbol{V}_t| \leq (Tr(\boldsymbol{V}_t)/K)^K$. We now look at the trace of $\boldsymbol{V}_t$:

$$Tr(\boldsymbol{V}_t) = Tr(\lambda \boldsymbol{I}_K) + Tr\left(\sum_{\tau=1}^{t} \boldsymbol{Z}_\tau^T \boldsymbol{Z}_\tau\right) = K\lambda + \sum_{\tau=1}^{t} Tr(\boldsymbol{Z}_\tau^T \boldsymbol{Z}_\tau) \tag{16}$$

$$\begin{aligned} Tr(\boldsymbol{Z}_\tau^T \boldsymbol{Z}_\tau) &= ||\boldsymbol{Z}_\tau||_F^2 = ||\boldsymbol{M}\boldsymbol{P}\boldsymbol{H}||_F^2 \leq Q\,||\boldsymbol{P}||_F^2\,||\boldsymbol{I}_K||_F^2\,||\boldsymbol{h}_\tau||_F^2 \\ &= KQ||\boldsymbol{P}||_F^2\,||\boldsymbol{h}_\tau||_F^2 \leq KNT_0||\boldsymbol{P}||_F^2 \end{aligned} \tag{17}$$

where the first inequality comes from $||\boldsymbol{M}||_F^2 \leq Q$, and the last inequality comes from the imposed sparsity level on $\boldsymbol{h}$ and from the condition $|h_n| \leq 1$. Finally, we look at the Frobenius norm of $\boldsymbol{P}$ as follows

$$||\boldsymbol{P}||_F^2 = \sum_k ||\Lambda^k||_F^2 = \sum_k \sum_l \lambda_l^k = d \tag{18}$$

where Λ is the diagonal matrix with entries the graph Laplacian eigenvalues λ_l. Thus, we can bound the trace of $\boldsymbol{V}_t$ as

$$Tr(\boldsymbol{V}_t) \leq K\lambda + tKQT_0d \tag{19}$$

and then derive the inequality of Lemma 1. □

Lemma 2: *Assume that* $\boldsymbol{V}_t = \sum_\tau \boldsymbol{Z}_\tau^T \boldsymbol{Z}_\tau + \lambda \boldsymbol{I}_K$*, define* $\boldsymbol{w}_\tau = \boldsymbol{Z}_\tau \boldsymbol{\alpha}_* + \boldsymbol{\eta}_\tau$*, with* $\boldsymbol{Z}_\tau = \boldsymbol{MPI}_K \otimes \boldsymbol{h}_\tau$ *and with* $\boldsymbol{\eta}_t$ *being conditionally R-sub-Gaussian, and assume that* $||\boldsymbol{\alpha}_*||_2 \leq S$*, and* $||\boldsymbol{h}_\tau||_F^2 \leq T_0$*. Then, for any* $\delta > 0$*, with probability at least* $1-\delta$*, for all* $t \leq 0$*,* $\boldsymbol{\alpha}_*$ *lies in the set*

$$E_t : \left\{ \boldsymbol{\alpha} \in \mathbb{R}^{1\times K} : ||\hat{\boldsymbol{\alpha}}_t - \boldsymbol{\alpha}||_{\boldsymbol{V}_t} \leq R\left[\sqrt{K\log(\lambda + tdQT_0) + 2\log(\lambda^{-1/2}\delta)}\right] + \lambda^{1/2}S \right\}$$

with $d = \sum_k \sum_l \lambda_l^k$*, with* λ_l *being the* l*-th eigenvalue of the graph Laplacian and* $\hat{\boldsymbol{\alpha}}_t$ *is the* l^2*-regularized least-square estimate of* $\boldsymbol{\alpha}_*$ *when* t *training samples are available.*

Proof: We now study the LOWER bound on the estimated $\hat{\boldsymbol{\alpha}}_t$, recalling that the vector is estimated observing the masked signal $\boldsymbol{w}_t$. Similarly to [1], using

$$\begin{aligned}\hat{\boldsymbol{\alpha}}_t &= \boldsymbol{V}_t^{-1}\boldsymbol{Z}_{1:t}^T\tilde{\boldsymbol{Y}}_{1:t}^T = \boldsymbol{V}_t^{-1}\boldsymbol{Z}_{1:t}^T\left(\boldsymbol{Z}_{1:t}\boldsymbol{\alpha}^\star + \boldsymbol{\eta}_{1:t}\right) = \boldsymbol{V}_t^{-1}\boldsymbol{Z}_{1:t}^T\boldsymbol{\eta}_t + \boldsymbol{V}_t^{-1}\boldsymbol{Z}_{1:t}^T\boldsymbol{Z}_{1:t}\boldsymbol{\alpha}^\star + \lambda\boldsymbol{V}_t^{-1}\boldsymbol{\alpha}^\star - \lambda\boldsymbol{V}_t^{-1}\boldsymbol{\alpha}^\star\\ &= \boldsymbol{V}_t^{-1}\boldsymbol{Z}_{1:t}^T\boldsymbol{\eta}_{1:t} + \boldsymbol{V}_t^{-1}\left(\boldsymbol{Z}_{1:t}^T\boldsymbol{Z}_{1:t} + \lambda\boldsymbol{I}_{K+1}\right)\boldsymbol{\alpha}^\star - \lambda\boldsymbol{V}_t^{-1}\boldsymbol{\alpha}^\star\\ &= \boldsymbol{V}_t^{-1}\boldsymbol{Z}_{1:t}^T\boldsymbol{\eta}_{1:t} + \boldsymbol{\alpha}^\star - \lambda\boldsymbol{V}_t^{-1}\boldsymbol{\alpha}^\star\,.\end{aligned} \tag{20}$$

Let's now consider a vector $x \in \mathbb{R}^{K\times 1}$, we get

$$\boldsymbol{x}^T\hat{\boldsymbol{\alpha}}_t - \boldsymbol{x}^T\boldsymbol{\alpha}^\star = \boldsymbol{x}\boldsymbol{V}_t^{-1}\boldsymbol{Z}_{1:t}^T\boldsymbol{\eta}_{1:t} - \lambda\boldsymbol{x}\boldsymbol{V}_t^{-1}\boldsymbol{\alpha}^\star = \langle\boldsymbol{x}, \boldsymbol{Z}_{1:t}^T\boldsymbol{\eta}_{1:t}\rangle_{V_t^{-1}} - \lambda\langle\boldsymbol{x}, \boldsymbol{\alpha}_*\rangle_{V_t^{-1}} \tag{21}$$

Using Cauchy-Schwarz, we get

$$\begin{aligned}|\boldsymbol{x}^T\hat{\boldsymbol{\alpha}}_t - \boldsymbol{x}^T\boldsymbol{\alpha}_*| &\leq ||\boldsymbol{x}||_{V_t^{-1}}\left(||\boldsymbol{Z}_{1:t}^T\boldsymbol{\eta}_{1:t}||_{V_t^{-1}} + \lambda||\boldsymbol{\alpha}_*||_{V_t^{-1}}\right)\\ &\leq ||\boldsymbol{x}||_{V_t^{-1}} R\sqrt{2\log\frac{|\boldsymbol{V}_t|^{1/2}}{\delta|\lambda\boldsymbol{I}_K|^{1/2}}} + \lambda^{1/2}S\end{aligned} \tag{22}$$

where the last inequality comes from Theorem 1 in [1], under the condition that η_t is conditionally R-sub-Gaussian for some $R \geq 0$, and that $||\boldsymbol{\alpha}_*||_2 \leq S$. In our case, $\boldsymbol{\eta} \sim \mathcal{N}(0, N\sigma_e^2)$, which means that it is a R-subgaussian variable with $R = \sqrt{N}\sigma_e$.

Using $\boldsymbol{x} = \boldsymbol{V}_t(\boldsymbol{\alpha}_t - \boldsymbol{\alpha}^\star)$ and $||\boldsymbol{V}_t(\boldsymbol{\alpha}_t - \boldsymbol{\alpha}^\star)||_{\boldsymbol{V}_t^{-1}} = ||\boldsymbol{\alpha}_t - \boldsymbol{\alpha}^\star||_{\boldsymbol{V}_t}$, we get

$$||\boldsymbol{\alpha}_t - \boldsymbol{\alpha}^\star||_{\boldsymbol{V}_t}^2 \leq ||\boldsymbol{V}_t(\boldsymbol{\alpha}_t - \boldsymbol{\alpha}^\star)||_{V_t^{-1}}\left(R\sqrt{2\log\frac{|\boldsymbol{V}_t|^{1/2}}{\delta|\lambda\boldsymbol{I}_{K+1}|^{1/2}}} + \lambda^{1/2}S\right). \tag{23}$$

Diving both sides by $||\boldsymbol{\alpha}_t - \boldsymbol{\alpha}^\star||$ we obtain

$$||\boldsymbol{\alpha}_t - \boldsymbol{\alpha}^\star||_{\boldsymbol{V}_t} \leq R\sqrt{2\log\frac{|\boldsymbol{V}_t|^{1/2}}{\delta|\lambda\boldsymbol{I}_{K+1}|^{1/2}}} + \lambda^{1/2}S \tag{24}$$

Applying Lemma 1, we obtain

$$||\hat{\boldsymbol{\alpha}}_t - \boldsymbol{\alpha}||_{\boldsymbol{V}_t} \leq R\left[\sqrt{K\log(\lambda + tdQT_0) + 2\log(\lambda^{-1/2}\delta)}\right] + \lambda^{1/2}S \tag{25}$$

This proves the confidence bound defined in Lemma 2. □

10 Proof of the Sub-Optimal Algorithm

In the following, we better motivate the Algorithm 2 proposed to solve (12). The problem formulation in (12) has a convex objective that needs to be maximized over a polytope. Because of the binary constraint on the $\boldsymbol{h}$ variable, the optimal solution will lie on one of the vertices of the boundary. Therefore, visiting all vertices would lead to the optimal solution. This however is not always practical in large graphs. Therefore, we propose an algorithm to jump from one vertex to the other in such a way that the objective function always increases and ideally the optimum point is reached within the maximum number of iteration allowed by the algorithm.

We propose an iterative procedure. At each iteration the algorithm selects one vertex among the ones neighboring to the node selected at the current iteration. Note that vertices are adjacent if they share all but one non-basic variable. We then consider the following Lemmas.

Lemma 3: *One vertex is optimal if there is no better neighboring vertex.*
Proof: This follows directly from the definition of convex function. Namely, the point of global optimum will be the one that maximizes the objective function. Therefore, moving out in any direction from the optimum point (*i.e.*, visiting any neighboring node) will not increase the objective function. At the same time, any other point which is not optimum will always have at least one neighboring node that leads to a no-worse objective function. □

The missing step is how to select the best direction to move from one iteration to the other. This is explained by the following Lemma.

Lemma 4: *From a vertex, moving to one of the neighboring nodes in the direction of the greatest gradient leads to a no-worse objective function. Let denote by $h_{in}^{(i)}$ and $h_{out}^{(i)}$ be the variable that enters and leaves the set of basic variables, respectively, at the i-th iteration. These variables are evaluated as follows*

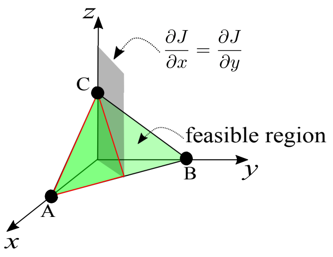
$$in = \arg\max_{n|h_n\notin\mathcal{B}_t}\left\{\frac{\partial J}{\partial h_n}\Big|_{\boldsymbol{h}^{(i-1)}}\right\}, \quad out = \arg\min_{n|h_n\in\mathcal{B}_t}\left\{\frac{\partial J}{\partial h_n}\Big|_{\boldsymbol{h}^{(i-1)}}\right\}$$

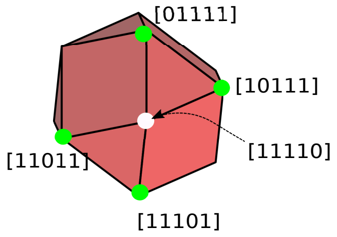
Proof: We first consider a simple case of $N = 3$ with the three possible solutions $\boldsymbol{h}_A = [1\,0\,0]$, $\boldsymbol{h}_B = [0\,1\,0]$, $\boldsymbol{h}_C = [0\,0\,1]$, see Fig. 6(a). Let also assume that $J(\boldsymbol{h}_C) < J(\boldsymbol{h}_B) < J(\boldsymbol{h}_A)$ and that $\boldsymbol{h}_C$ is the starting node. The algorithm needs to decide if moving in the direction of $\boldsymbol{h}_B$ and $\boldsymbol{h}_A$. From the definition of gradient, the gradient of J evaluated in $\boldsymbol{h}_C$ will be pointing toward $\boldsymbol{h}_A$ rather than $\boldsymbol{h}_B$ (since the improvement of the objective function if larger is moving in this direction). Let make the approximation of having a continuous domain (rather than a discrete one) highlgihted by the green shaded area in Fig. 6(a) and assume to run the gradient ascend to find the optimal solution. The gradient from $\boldsymbol{h}_C$ will be pointing to any region highlighted in red in Fig. 6(a) since at last the gradient ascend will be point toward $\boldsymbol{h}_A$. This that that the variable at the iteration $i+1$ would be

$$\boldsymbol{h}^{(i+1)} = \boldsymbol{h}_C + \alpha \left(\frac{\partial J}{\partial x}\boldsymbol{i} + \frac{\partial J}{\partial x}\boldsymbol{j} + \frac{\partial J}{\partial x}\boldsymbol{k} \right).$$

and to move toward $\boldsymbol{h}_A$, the following condition needs to be respected

$$\frac{\partial J}{\partial x} \geq \frac{\partial J}{\partial y}$$


z
$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial y}$
C
feasible region
B
y
A
x
(a) graphical representation



(b) pseudo-code

Fig. 6. Visualization of the feasible region for a three dimensional domain. (a) Red contour delimit the area in which the gradient will be pointing to. (b) white circle denotes the starting node (or vertex). Green vertices are the candidate ones.

This condition can be extended to a more general case with a starting node $\boldsymbol{h}^{(i)} = [h_1^{(i)}, h_2^{(i)}, \ldots, h_N^{(i)}]$ at the i-th iteration. Without loss of generality we assume that the first T_0 variables are the basic variables, this means

$$\boldsymbol{h}^{(i)} = [\underbrace{h_1^{(i)}, h_2^{(i)}, \ldots, h_{T_0}^{(i)}}_{\text{basic variables}}, \underbrace{h_{T_0+1}^{(i)}, \ldots, h_N^{(i)}}_{\text{non-basic variables}}] = [\underbrace{1, 1, \ldots, 1}_{\text{basic variables}}, \underbrace{0, \ldots, 0}_{\text{non-basic variables}}]$$

Since neighboring nodes differs in the non-basic variables only for one element, it means that only one of the $N - T_0$ remaining variables can become a basic variable. Let consider that the candidates neighboring vertices are

$$
\begin{aligned}
\boldsymbol{h}_a &= [\underbrace{0,1,\dots,1}_{T_0},\underbrace{1,0,\dots,0}_{N-T_0}]\\
\boldsymbol{h}_b &= [\underbrace{0,1,\dots,1}_{T_0},\underbrace{0,1,\dots,0}_{N-T_0}]\\
&\dots\\
\boldsymbol{h}_l &= [\underbrace{0,1,\dots,1}_{T_0},\underbrace{0,0\dots,1}_{N-T_0}]
\end{aligned}
$$

This means that we simply need to select the direction out of $N - T_0$ that maximizes the gain in terms of objective function. This translates in the following condition

$$\max_{n|h_n\notin\mathcal{B}_t}\left\{\frac{\partial J}{\partial h_n}\Big|_{\boldsymbol{h}^{(i)}}\right\}. \tag{26}$$

The above condition determines a T_0-dimensional plane with T_0possible vertices (green vertices in Fig. 6(b)). To select the best one, we identify the least promising direction among the ones of the basic variables in $\boldsymbol{h}^{(i)}$. This translates in the second condition imposed in Algorithm 2, which is

$$\min_{n|h_n\in\mathcal{B}_t}\left\{\frac{\partial J}{\partial h_n}\Big|_{\boldsymbol{h}^{(i-1)}}\right\} \tag{27}$$

The motivation is the symmetric one of above, but this time we look for the least convenient direction since we look for the variable to abandon the set of basic variables. Therefore, we seek the direction that minimizes the gradient. □

11 Optimal Solution for (7)

The problem in (12) can also be solved as follows. We relax the sparsity constraint with a $l1$-norm constraint, leading to the following equivalent problem

$$
\begin{aligned}
&\max_{\boldsymbol{h}} \boldsymbol{D}\boldsymbol{h} + c_t||\boldsymbol{L} * \boldsymbol{b}^T \otimes \boldsymbol{h}^T||_2\\
&\text{s.t. } h(n) \in [0,1],\ \forall n\\
&\qquad ||\boldsymbol{h}||_1 = \sum_{n=1}^{N} h(n) \le T_0
\end{aligned} \tag{28}
$$

The above problem is a maximization of a convex problem over a polytope and can be globally solved by solvers for constrained nonlinear optimization problems, as branch-and-bound search (*e.g.*, bmibnb in Matlab) or sequential quadratic programming (SQP). The branch-and-bound solver is based on a spatial branch-and-bound strategy, based on linear programming relaxations and convex envelope approximations. Relaxed problems are solved to evalute lowe and upper bounds. Given these lower and upper bounds, a standard branch-and-bound logic is used to select a branch variable, create two new nodes, branch, prune and navigate among the remaining nodes. On the other side, the sequential quadratic programming is an iterative procedure which models the originating problem by a quadratic programming subproblem at each iteration. In both cases, the problem stays NP-hard and the solvers require exponential computational complexity to achieve the optimal solution. Specifically, it can be observed that the cardinality of the decision variables $|\mathcal{A}|$ is $\binom{N}{T_0}$ and the complexity of the solver grows exponentially with $|\mathcal{A}|$.

12 UCB Derivation

By least square regression, we estimate the mean reward, but we can also estimate the variance of reward, denoted by σ_α^2, i.e. the uncertainty due to parameter estimation error. We can then define the UCB to be c standard deviations above the mean by adding on a bonus for uncertainty, $c\sigma_\alpha$ to the mean reward. This leads to maximize the reward summed on the UCB.

Being a linear regression, the parameter covariance is V_t^{-1}. being the reward linear in features ($\boldsymbol{X\alpha}$), we obtain a quadratic reward variance $\boldsymbol{X}^T V_t^{-1}\boldsymbol{X}$. The geometric interpretation is that we maximize the reward for any parameter vector α within an ellipsoid E_t defined by c. It follows

$$\begin{aligned}
\boldsymbol{h}_t &= \arg\max_{\boldsymbol{h}\in\mathcal{A}} \max_{\alpha\in E_t} \boldsymbol{X\alpha} \\
&= \arg\max_{\boldsymbol{h}\in\mathcal{A}} \left(\boldsymbol{X\alpha} + c_t\sigma_{X\alpha}\right) \\
&= \arg\max_{\boldsymbol{h}\in\mathcal{A}} \left(\boldsymbol{X\alpha} + c_t\sqrt{\boldsymbol{X}\boldsymbol{V}_t^{-1}\boldsymbol{X}^T}\right) \\
&= \arg\max_{\boldsymbol{h}\in\mathcal{A}} \boldsymbol{X\alpha} + c_t||\boldsymbol{X}||_{\boldsymbol{V}_t^{-1}} \\
&= \arg\max_{\boldsymbol{h}\in\mathcal{A}} \left[\boldsymbol{MPH}\hat{\boldsymbol{\alpha}}_t + c_t||\boldsymbol{MPH}||_{\boldsymbol{V}_t^{-1}}\right] \qquad (29)
\end{aligned}$$

with $c_t = R\left[\sqrt{K\log(\lambda + tdQT_0) + 2\log(\lambda^{-1/2}\delta)}\right] + \lambda^{1/2}S$ following Lemma 2.

13 Additional Experimental Results

13.1 Influence of the Graph Topology

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 better the decision maker behaviour. 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.

The key intuition we have from Lemma 2 is that the confidence bound (and therefore the uncertainty on the estimation) increases with the sparsity level T_0 as well as the parameter d, which represents the power sum of the eigenvalue of the Laplacian[6]. Specifically,

$$d = \sum_{k=0}^{K}\sum_{l=1}^{N}\lambda_l^k \leq \tilde{N}\sum_{k=0}^{K}\lambda_{\max}^k = \tilde{N}\frac{1-\lambda_{\max}^{k+1}}{1-\lambda_{\max}} \tag{30}$$

where $\tilde{N}$ is the number of eigenvalues considerably larger than 0, and $\lambda_{\max}$ the maximum eigenvalue of the graph Laplacian. Note that the last equality holds from the geometric series.

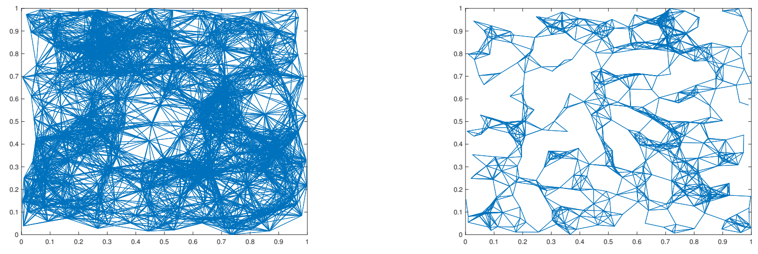
We are therefore interested in studying how much the estimation error depends on the connectivity of the graph, and therefore on the Laplacian. We consider graphs generated with the RBF model. By changing the threshold parameter T, we generate more or less connected graphs. In Fig. 7, the graph topologies for $T = 0.95$ and $T = 0.987$ when $N = 400$ is provided. As expected, the smaller T is, the more connected is the graph. Higher levels of connectivity also leads to a more narrow profile of the eigenvalues of the Laplacian λ_l, as observed from Fig. 8(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, for each graph topology. As a consequence, more connected graphs lead to a more accurate estimate of the generating kernels, see Fig. 8(b). The intuition is that more connected graphs lead 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. 8(a)) and the associated power sum d. Similar behavior is observed in the case 100 nodes, as depicted in Fig. 9.

Similar results are observed in the case of scale-free graphs, as the ones generated with the BA model. Fig. 10 depicts two graph topologies with different m, which is the number of nodes with which a new node in the network connects to. Therefore, the larger the m the more connected is the graph. Fig. 11 confirms the behaviors already observed with the RBF model. Namely, more connected graphs permit to learn better.

13.2 Influence of source sparsity

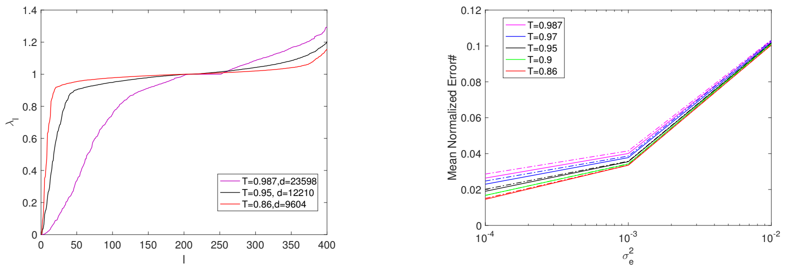
In Fig. 12, we then evaluate the mean normalized error as a function of the sparsity level of the source signal. {**PF:** *maybe we can add a sentence of motivation for this analysis.*} Results are plotted as a function of the normalized sparsity

[6] Note that the estimation error increases also with variance of the random noise $\boldsymbol{\eta}$ but this is not related to the graph topology, therefore it is beyond the scope of this section.



(a) High connected graph (T=0.95) (b) Low connected graph (T=0.987)

Fig. 7. Different graph topologies in the case of $N = 400$ nodes, $\sigma = 0.5$ and we vary $T = 0.95$ and $T = 0.987$.



(a) Graph Laplacian Eigenvalues (b) Estimation Error

Fig. 8. Graph Laplacian distribution and signal estimation error for random graphs with different levels of connectivity, $N = 400$ nodes, and sparsity value $T_0 = 25$.The estimation error is evaluated both in the case of full observability (solid line) or in the case of partial observability (dotted line) with a mask covering 40% of the nodes.

level, defined as T_0/N, and they are provided for a random graph with 100 nodes (dashed line) and with 400 nodes (solid line). Finally, a diffusion process with $\tau = 10$ is considered and estimated by the polynomial $\boldsymbol{\alpha}$ of degree $K = 20$. Simulations are considered for different noise variances, namely 10^{-3} and 10^{-2}, which are compared to the noiseless case. It is interesting to observe that larger sparsity levels lead to an increase of the estimation error. This is motivated by the following observation: less source signals (*i.e.*, smaller sparsity value T_0) lead to more localized information so that sources are more informative. Therefore, a more sparse signal $\boldsymbol{h}$ leads to a better estimation. This can also be demonstrated by Lemma 2, where mathematically it can be observed that the confidence bound is proportional to the sparsity value T_0. Therefore the higher T_0 the greater the uncertainty about the estimation. Finally, from Lemma 2 the estimation error should increases also with the number of nodes N, while in Fig. 12 we observe the opposite behavior. This is because the number of nodes increases but within

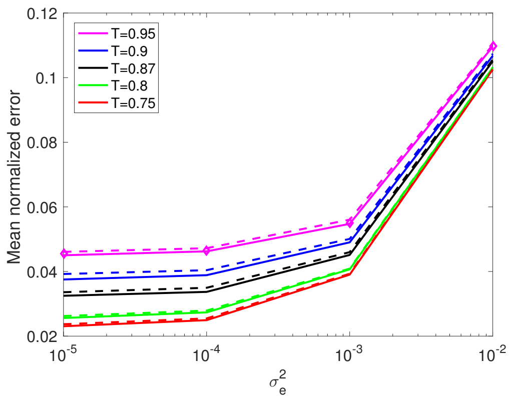


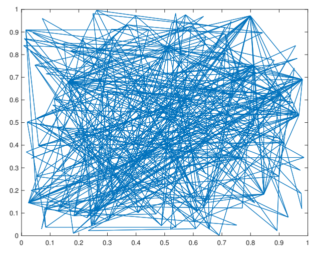
Fig. 9. Signal estimation error vs noise variance for random graphs with different levels of connectivity, $N = 100$ nodes, and sparsity value $T_0 = 15$.The estimation error is evaluated both in the case of full observability (solid line) or in the case of partial observability (dotted line) with a mask covering 40% of the nodes.

the same unitary space (varying the density of the graph) and it is observed that denser graphs ($N = 400$) outperform less dense ones ($N = 100$). This is in line with the above observation that more connected (and more dense) graphs lead to a better estimate.

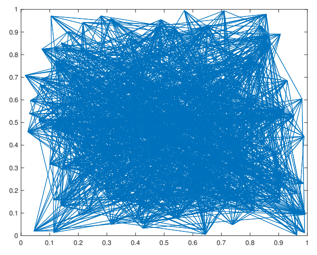
14 Bandits Overview

14.1 Online Learning

{**PF:** *+general introduction sentence that recall settings + problem + motivation for bandits*} In the classical stochastic k-armed bandit problem {**PF:** *+ ref*}, at time t, the agent (or the learner) selects an action $a_t \in \mathcal{A}$, where $\mathcal{A}$ is the action set with cardinality $|\mathcal{A}| = k$. A non-negative mean reward is associated to each arm following the mapping $r(a_t) : \mathcal{A} \to \mathbb{R}$. This mapping is unknown *a priori* by the agent that only observes the stochastic instantaneous reward $w_t = r(a_t)+n_t$, with n_t being a subgaussian random noise. This agent-environment game (that reveals the reward) is played over T rounds, where T is a positive natural number called the horizon and $t \in [0, T]$. The learner goal is to choose sequential actions that maximize the cumulative reward over all T rounds, i.e., $\sum_{t=1}^{T} w_t$. Equivalently, it seeks to minimize the cumulative regret (i.e., cumulative loss incurred by not selecting the optimal action $a^\star$) given by $R_T = T\, r(a^\star) - \mathbb{E}\left(\sum_{t=1}^{T} r(a_t)\right)$, which corresponds to optimizing the pseudo-regret [18], as commonly done in the MAB literature. When no prior assumption is made on the structure of the process to be learned, the agent can only blindly infer the mapping $a_t \to r\,(a_t)$ via the well-known UCB (upper confidence bound) algorithm [18]. This achieves

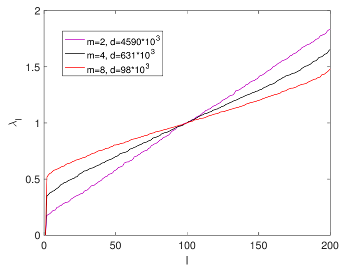


(a) High connected graph (m_0 = 2)

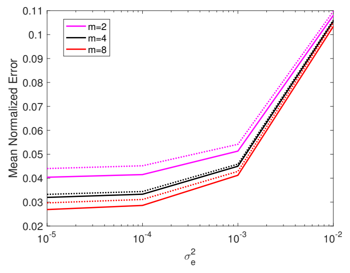


(b) Low connected graph ($m_0 = 8$)

Fig. 10. Different graph topologies generated with the BA model in the case of $N = 300$ nodes, $m_0 = 2$ and $m_0 = 8$.



(a) Graph Laplacian Eigenvalues



(b) Estimation Error

Fig. 11. Graph Laplacian distribution and signal estimation error for graphs generated with the BA model with different levels of connectivity, $N = 200$ nodes, and sparsity value $T_0 = 25$. The estimation error is evaluated both in the case of full observability (solid line) or in the case of partial observability (dotted line) with a mask covering 40% of the nodes.

a sublinear regret, *i.e.*, $R_T = \mathcal{O}(|\mathcal{A}| \log T)$, with $|\mathcal{A}|$ being the cardinality of the action space, leading to highly supoptimal network optimization in the case of high-dimensional action sets like those on large graphs.

In most of the cases however, the learner is able to exploit some prior knowledge on the environment, i.e., at time t the learner has access to a context $c_t \in \mathcal{C}$, which provides insights about the reward mapping. For example, in the case of recommendation systems for online purchases, the context could be prior information about the items to sell or about the buyers. A widely used setting is the linear stochastic (contextual) bandit in which each action at time t is identified by the feature vector $\boldsymbol{x}_t = \psi(a_t, c_t)$ with $\boldsymbol{x}_t \in \mathbb{R}^{d\times 1}$, and with $\psi : \mathcal{A} \times \mathcal{C} \to \mathbb{R}^{d\times 1}$ being the contextual mapping function. In this case, the reward is a noisy linear

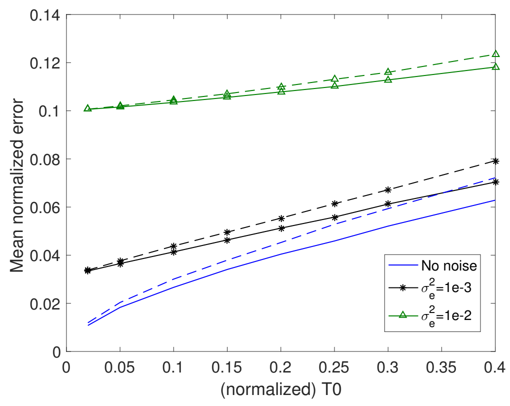


Fig. 12. Mean normalized error vs the sparsity level of the source signal for graphs with 100 (dashed line) and 400 (solid line) nodes in the case of full observability.

function given by

$$w_t = \boldsymbol{x}_t^T \boldsymbol{\theta} + n_t \tag{31}$$

with $\boldsymbol{\theta} \in \mathbb{R}^{d \times 1}$ being an unknown parametric vector that the agent needs to learn over time. When the prior knowledge related to the action is separated from any other context information, we obtain instead a reward given by $w_t = \boldsymbol{x}_t^T \boldsymbol{\theta}_{i_t} + n_t$. In this case, the arm feature vector only captures key information about the arms, $\boldsymbol{x}_t = \psi(a_t)$, with $\psi : \mathcal{A} \rightarrow \mathbb{R}^{d \times 1}$, and the unknown parameter $\boldsymbol{\theta}_{i_t}$ captures information about the context identified by the index i_t. This is usually the case of multi-user bandit problems {**PF:** + *ref*}.

In the above cases, the linear reward allows us to apply the linear UCB (upper confidence bound) algorithm, namely LinUCB [10], to select actions in such a way that the exploration and exploitation phases are well balanced in the online learning problem. In short, in LinUCB the agent selects the actions with the highest reward, inflated by the upper confidence bound. The latter is evaluated as the uncertainty of the least square estimator that aims at inferring the unknown parameter $\boldsymbol{\theta}$ in the linear reward. Given the linear reward in (31), for example, the agent at time t has collected past experience $\{\boldsymbol{x}_\tau, w_\tau\}_{\tau=0}^t$ from which it estimates the unknown vector $\hat{\theta}_t$ via regularized least square estimation. The uncertainty on the estimated vector is measured by the ellipsoid E_t centered in $\hat{\theta}_t$, which contains the ground truth θ with probability $1-\delta$. The agent then chooses the best action in the best environment amongst the plausible ones (the ones in the confidence ellipsoid)

$$\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}} \max_{\boldsymbol{\theta} \in E_t} \langle \boldsymbol{x}, \boldsymbol{\theta} \rangle = \arg\max_{\boldsymbol{x}} \langle \boldsymbol{x}, \hat{\boldsymbol{\theta}}_t \rangle + \underbrace{\beta_t ||\boldsymbol{x}||_{\boldsymbol{V}_{t-1}^{-1}}}_{\mathrm{U}(E_t)} \tag{32}$$

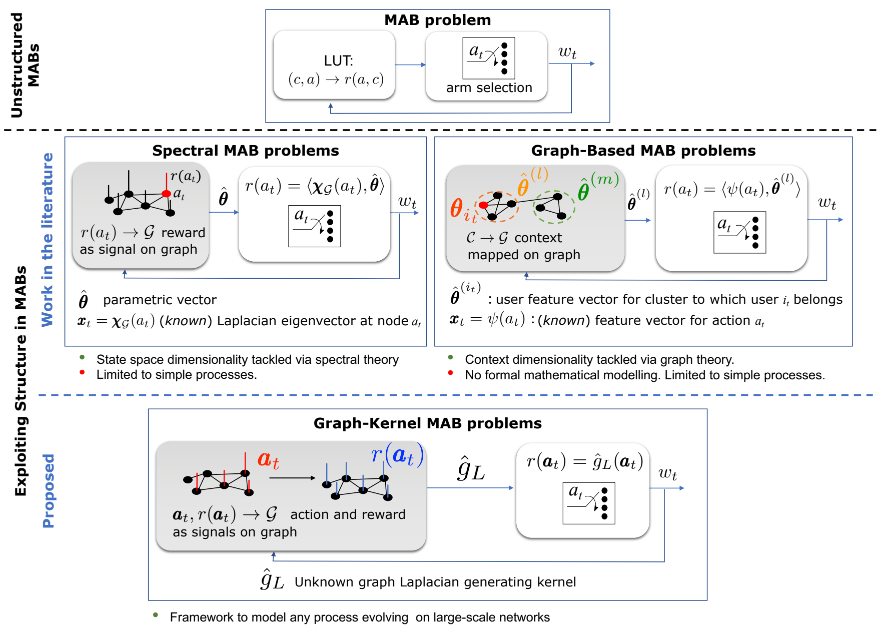

Unstructured MABs
MAB problem
LUT: $(c,a) \rightarrow r(a,c)$
arm selection
w_t
Exploiting Structure in MABs
Work in the literature
Spectral MAB problems
$r(a_t) \rightarrow \mathcal{G}$ reward as signal on graph
$\hat{\boldsymbol{\theta}}$
$r(a_t) = \langle \boldsymbol{\chi}_{\mathcal{G}}(a_t), \hat{\boldsymbol{\theta}} \rangle$
$\hat{\boldsymbol{\theta}}$ parametric vector
$\boldsymbol{x}_t = \boldsymbol{\chi}_{\mathcal{G}}(a_t)$ (known) Laplacian eigenvector at node a_t
State space dimensionality tackled via spectral theory
Limited to simple processes.
Graph-Based MAB problems
$\mathcal{C} \rightarrow \mathcal{G}$ context mapped on graph
$r(a_t) = \langle \psi(a_t), \hat{\boldsymbol{\theta}}^{(l)} \rangle$
$\hat{\boldsymbol{\theta}}^{(i_t)}$: user feature vector for cluster to which user i_t belongs
$\boldsymbol{x}_t = \psi(a_t)$: (known) feature vector for action a_t
Context dimensionality tackled via graph theory.
No formal mathematical modelling. Limited to simple processes.
Proposed
Graph-Kernel MAB problems
$\boldsymbol{a}_t, r(\boldsymbol{a}_t) \rightarrow \mathcal{G}$ action and reward as signals on graph
$\hat{g}_L$
$r(\boldsymbol{a}_t) = \hat{g}_L(\boldsymbol{a}_t)$
$\hat{g}_L$ Unknown graph Laplacian generating kernel
Framework to model any process evolving on large-scale networks

Fig. 13. Graphical visualisation of structured and unstructured MAB problems.

where the last equality comes from the maximization of a linear function over a convex closed set [6], $\boldsymbol{V}_{t-1} = \sum_{\tau=0}^{t} \boldsymbol{x}_t \boldsymbol{x}_t^T$, and β_t identifies the size of the confidence ellipsoid at t. In short, the agent selects the action with the highest UCB index, which is the estimated mean reward inflated of a exploration bonus term $\mathrm{U}(E_t)$ that favors exploration in the case of large uncertainty. LinUCB is widely adopted in small scale settings, however it becomes highly inefficient when the search space $\mathcal{A} \times \mathcal{C}$ is high dimensional, as in most of real-world use cases. Such limitation can be overcome by exploiting the structure of the context [33, 39].

14.2 Structured Bandits

In online learning in high dimensions, it becomes important to exploit the structure of the problem and of the context in order to develop effective solutions. In [7, 8, 12, 17, 20–22], the multi-user bandit problem is considered, and the affinity between users is encoded by an undirected and weighted graph $\mathcal{G}$ (see Fig. 13, box "**Graph-Based MAB problems**"). Graph clustering is a possible solution to reduce the dimensionality of the problem while preserving the performance of the learning system [12, 20, 21], leading to an estimated context feature $\hat{\boldsymbol{\theta}}^{(l)}$ for the cluster l to which the user i_t belongs. The learned performance however depends on the clustering algorithm being used, which tends to be expensive for large-scale graphs, and does not necessarily perform well in realistic datasets [44].

Another approach is the one in which each arm reward is shared across neighbouring nodes within a graph [19, 23]. The regret bounds are then derived as a function of the graph structure. However, the geometric information of the problem is not taken into account during the arm selection, resulting in suboptimal strategies.

Rather than exploiting the structure of the context, recent works have studied the structure of the reward signal, looking at the arms as nodes on a graph $\mathcal{G}$ and the reward as a smooth signal on this graph [13, 16, 37, 38] (see Fig. 13, box "**Spectral MAB problems**"). Differently from the above Graph-Based MAB problems, the geometrical structure is now imposed on the observed domain (i.e., reward domain) instead of the context domain. This permits to define the reward as a linear combination of the eigenvectors of the graph Laplacian matrix, where the linear coefficients are unknown. When applying LinUCB [10] in the spectral domain, one can learn the unknown coefficients and estimate the mean reward associated to an action (represented as one node on the graph) a_t as a function of these coefficients and the spectral graph representation given by its eigenvectors $\boldsymbol{\chi}_{\mathcal{G}}(a_t)$. These algorithms achieve a regret bound of the order $\sqrt{dT}$, with d being the effective dimension (linked to the dimensionality of the characteristic eigenvalues) and T being the number of rounds. Similar intuitions have been introduced in [36], which performs maximization over the smooth functions that have a small Reproducing kernel Hilbert space (RKHS) norm, or in [43] for a multi-user bandit in which the user features are modelled as a smooth signal on a graph. These settings are also more general since they generalize linear bandits. However, the spectral MAB problem relies on the following main assumptions: 1) the reward signal is smooth on the graph; 2) the graph represents the action domain, i.e., one action is associated to one single node on the graph and the reward is the signal evaluated at that given node. This limits the applications of spectral MAB problems, preventing online network optimizations in which 1) each action represents possibly a set on a graph (not limited to a node only) and the reward, not necessarily smooth on the graph, is a resultant signal on the entire graph – not necessarily the signal on one node only. Inspired by spectral MAB problems. We aim at developing a more general framework to be applied to a more general set of problems, namely online optimization of non-trivial processes evolving over large-scale networks.

In the following, we formalize the online learning problem, given these two key properties.

15 Additional pieces of text

15.1 Online Learning Solution

For the agent to select the optimal action maximizing the mean reward, the graph generating kernel needs to be learned. It is learned given the dataset $\{\boldsymbol{a}_\tau, \boldsymbol{w}_\tau\}_{\tau=0}^{t}$ collected by the agent over time, leading to the well known exploitation-exploitation tradeoff. In the case of linear mapping between the unknown parameters $\boldsymbol{\alpha}$ and

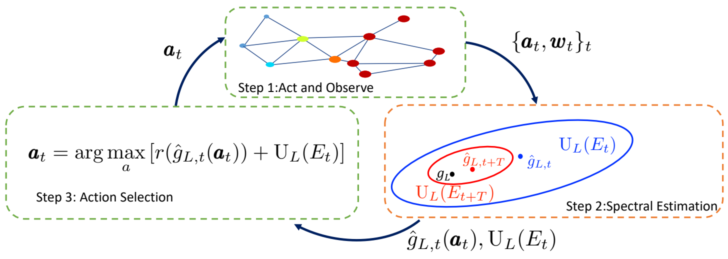

$\boldsymbol{a}_t$
Step 1:Act and Observe
$\{\boldsymbol{a}_t, \boldsymbol{w}_t\}_t$
$\boldsymbol{a}_t = \arg\max_a [r(\hat{g}_{L,t}(\boldsymbol{a}_t)) + \mathrm{U}_L(E_t)]$
Step 3: Action Selection
$\mathrm{U}_L(E_t)$
$\hat{g}_{L,t+T}$
$\hat{g}_{L,t}$
g_L
$\mathrm{U}_L(E_{t+T})$
Step 2:Spectral Estimation
$\hat{g}_{L,t}(\boldsymbol{a}_t), \mathrm{U}_L(E_t)$

Fig. 14. Figurative example of the online graph-strcutured processing. Green (red) dashed boxes are defined in the vertex (spectral) domain.

Table 1. Notation for Graph-Kernel MAB problems.

	MAB Notation	GSP Notation
action	$\boldsymbol{a}$	$\boldsymbol{h}$ (or $\boldsymbol{\alpha}$)
mean reward	$r(\boldsymbol{a})$	$f\left(g_{\boldsymbol{L}}(\boldsymbol{h};\boldsymbol{\alpha})\right) = \boldsymbol{X}\boldsymbol{\alpha}$
instantaneous reward	$\boldsymbol{w} = r(\boldsymbol{a}) + \boldsymbol{n}$	$\boldsymbol{w} = \boldsymbol{M}\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\alpha} + \boldsymbol{n}$

the actions $\boldsymbol{h_t}$ (hence $\boldsymbol{X}_t$), the tradeoff is achieved in following the linUCB algorithm (32), which in our case is formalized as follow

$$\boldsymbol{h}_t = \arg\max_{\boldsymbol{h}} \left[\boldsymbol{X}_t \hat{\boldsymbol{\alpha}}_t + \mathrm{U}_L(E_t)\right] . \tag{33}$$

where $\hat{\boldsymbol{\alpha}}_t$ is the estimate of $\boldsymbol{\alpha}$ given the dataset collected up to time t.

With our proposal, we observe that the reward is evaluated in the vertex domain, while the confidence bound E_t is computed in the spectral domain. We recall that E_t represents the uncertainty on the estimation of $\boldsymbol{\alpha}$, which represents the graph kernel, defined in the spectral domain. Specifically, E_t represents the ellipsoid centered in $\hat{\boldsymbol{\alpha}}_t$ containing the ground truth $\boldsymbol{\alpha}$ with $1-\delta$ probability {**PF:** *clear? ref or pointer? with the different change, we don't talk much about E_t before that point - maybe we can explain in two words what it is used for in the RL algo*}. Rather than acting and learning in a high-dimensional domain, we use GSP tools to act in the vertex domain and learn in the spectral domain, which is of lower dimensionality, see Fig. 14. This presents an important advantage, as the regret scales as $\mathcal{O}(d\sqrt{T}\log T)$ in LinUCB [10], where d is the dimension of the unknown parameter $\boldsymbol{\alpha}$ (i.e., d has the dimension of K in our case, which is much smaller than the search space $|\boldsymbol{h}|$ {**PF:** *is that the correct notation? other clearer way to denote the dimension of the search space?*}).

In the next Sections, we consider a specific graph-kernel MAB problem, and derive the corresponding bounds, which permit to eventually validate the benefits of the proposed methods in a series of experiments.

Algorithm 3 Act After Learning (AAL)

```
Input:
$N$: nr of nodes, $T_0$: sparsity level of initial signal $\boldsymbol{h}$, $K$: sparsity of the basis coefficients
$\lambda, \delta$: regularization and confidence parameters
while $t \leq T_L$ do
    Select the action $\boldsymbol{h}_t$ (and therefore $\boldsymbol{x}_t$) uniformly at random among the all possible
    actions
    Take action $\boldsymbol{h}(T_L)$ and observe the reward $y_t$
end while
Estimate the generating kernel:
$\boldsymbol{X}_{1:T_L} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{T_L-1}]^T$
$\boldsymbol{Y}_{1:T_L} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{T_L-1}]^T$
$\boldsymbol{V}_{T_L} = \boldsymbol{X}_{1:T_L}^T \boldsymbol{X}_{1:T_L} + \lambda \boldsymbol{I}_{K+1}$
$\hat{\boldsymbol{\alpha}}_{T_L} = \boldsymbol{V}_{T_L}^{-1} \boldsymbol{X}_{1:T_L}^T \boldsymbol{Y}_{1:T_L}$
Select the best action based on the estimated generating kernel:
$\boldsymbol{h}(T_L) : \arg\max_{\boldsymbol{h} \in \mathcal{A}} [\mathbf{1}_N \boldsymbol{P}\boldsymbol{H}\hat{\boldsymbol{\alpha}}_t]$
$t = T_L + 1$
while $t \leq T$ do
    Refine estimate of the coefficients
    Take action $\boldsymbol{h}(T_L)$ and observe the reward $y_t$
    $t = t + 1$
end while
```

References

1. Abbasi-yadkori, Y., Pál, D., Szepesvári, C.: Improved algorithms for linear stochastic bandits. In: Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., Weinberger, K.Q. (eds.) nips, pp. 2312–2320 (2011)
2. Acemoglu, D., Ozdaglar, A.: Opinion dynamics and learning in social networks. Dynamic Games and Applications **1**(1), 3–49 (Mar 2011)
3. Avner, O., Mannor, S.: Multi-user communication networks: A coordinated multi-armed bandit approach. IEEE/ACM Transactions on Networking **27**(6), 2192–2207 (2019)
4. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
5. Bellemare, M.G., Dabney, W., Dadashi, R., Taïga, A.A., Castro, P.S., Roux, N.L., Schuurmans, D., Lattimore, T., Lyle, C.: A geometric perspective on optimal representations for reinforcement learning. CoRR **abs/1901.11530** (2019)
6. Boyd, S., Boyd, S.P., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)
7. Caron, S., Kveton, B., Lelarge, M., Bhagat, S.: Leveraging side observations in stochastic bandits. ArXiv **abs/1210.4839** (2012)
8. Cesa-Bianchi, N., Gentile, C., Zappella, G.: A gang of bandits. In: nips. pp. 737–745 (2013)
9. Cesa-Bianchi, N., Lugosi, G.: Prediction, learning, and games. Cambridge university press (2006)
10. Chu, W., Li, L., Reyzin, L., Schapire, R.E.: Contextual bandits with linear payoff functions. In: Proc. Artificial Intelligence and Statistics Conference (AISTATS). vol. 15, pp. 208–214 (2011)

11. Gentile, C., Herbster, M., Pasteris, S.: Online similarity prediction of networked data from known and unknown graphs. In: Conference on Learning Theory. pp. 662–695 (2013)
12. Gentile, C., Li, S., Zappella, G.: Online clustering of bandits. In: Proc. International Conference on Machine Learning (ICML) (2014)
13. Hanawal, M.K., Saligrama, V.: Cost effective algorithms for spectral bandits. In: Proc. IEEE Conf. on Communication, Control, and Computing (2015)
14. He, Y., Wai, H.T.: Detecting central nodes from low-rank excited graph signals via structured factor analysis. arXiv preprint arXiv:2109.13573 (2021)
15. Jones, N.: How to stop data centres from gobbling up the world's electricity. Nature **561**(7722), 163–167 (2018)
16. Kocák, T., Valko, M., Munos, R., Agrawal, S.: Spectral thompson sampling. In: Proc. AAAI Conf. on Artificial Intelligence (2014)
17. Korda, N., Szorenyi, B., Li, S.: Distributed clustering of linear bandits in peer to peer networks. In: Proc. International Conference on Machine Learning (ICML) (2016)
18. Lattimore, T., Szepesvári, C.: Bandit algorithms. arXiv (2018)
19. Lee, C.W., Luo, H., Zhang, M.: A closer look at small-loss bounds for bandits with graph feedback. In: in Proc. International Conference on Algorithmic Learning Theory (ALT) (2020)
20. Li, S., Gentile, C., Karatzoglou, A., Zappella, G.: Data-dependent clustering in exploration-exploitation algorithms. arXiv preprint arXiv:1502.03473 (2015)
21. Li, S., Gentile, C., Karatzoglou, A., Zappella, G.: Online context-dependent clustering in recommendations based on exploration-exploitation algorithms. ArXiv **abs/1608.03544** (2016)
22. Li, S., Karatzoglou, A., Gentile, C.: Collaborative filtering bandits. In: Proc. Int. ACM Conf. on Research and Development in Information Retrieval (2016)
23. Lykouris, T., Tardos, E., Wali, D.: Feedback graph regret bounds for thompson sampling and ucb. In: in Proc. International Conference on Algorithmic Learning Theory (ALT) (2020)
24. Mohaghegh Neyshabouri, M., Gokcesu, K., Gokcesu, H., Ozkan, H., Kozat, S.S.: Asymptotically optimal contextual bandit algorithm using hierarchical structures. IEEE Transactions on Neural Networks and Learning Systems **30**(3), 923–937 (2019)
25. Movric, K.H., Lewis, F.L.: Cooperative optimal control for multi-agent systems on directed graph topologies. IEEE Transactions on Automatic Control **59**(3), 769–774 (March 2014)
26. Nassif, R., Vlaski, S., Sayed, A.H.: Adaptation and learning over networks under subspace constraints. ArXiv **1905.08750** (2019)
27. Ortega, A., Frossard, P., Kovačević, J., Moura, J.M., Vandergheynst, P.: Graph signal processing: Overview, challenges, and applications. Proceedings of the IEEE **106**(5), 808–828 (2018)
28. Perra, N., Rocha, L.E.: Modelling opinion dynamics in the age of algorithmic personalisation. Scientific reports **9**(1), 1–11 (2019)
29. Ramakrishna, R., Scaglione, A.: Grid-graph signal processing (grid-gsp): A graph signal processing framework for the power grid. IEEE Transactions on Signal Processing **69**, 2725–2739 (2021)
30. Ramakrishna, R., Scaglione, A.: Grid-graph signal processing (grid-gsp): A graph signal processing framework for the power grid. IEEE Transactions on Signal Processing **69**, 2725–2739 (2021). https://doi.org/10.1109/TSP.2021.3075145

31. Salami, H., Ying, B., Sayed, A.H.: Social learning over weakly connected graphs. IEEE Transactions on Signal and Information Processing over Networks **3**(2), 222–238 (June 2017)
32. Shahrampour, S., Rakhlin, A., Jadbabaie, A.: Multi-armed bandits in multi-agent networks. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2786–2790 (2017)
33. Slivkins, A.: Contextual bandits with similarity information. Journal of Machine Learning Research **15**(1), 2533–2568 (2014)
34. Tang, S.: When social advertising meets viral marketing: Sequencing social advertisements for influence maximization. In: AAAI (2018)
35. Thanou, D., Shuman, D.I., Frossard, P.: Learning parametric dictionaries for signals on graphs. IEEE Trans. on Signal Processing **62**(15), 3849–3862 (2014)
36. Valko, M., Korda, N., Munos, R., Flaounas, I., Cristianini, N.: Finite-time analysis of kernelised contextual bandits (2013)
37. Valko, M., Munos, R.: Cheap bandits. In: Proc. International Conference on Machine Learning (ICML) (2015)
38. Valko, M., Munos, R., Kveton, B., Kocak, T.: Spectral bandits for smooth graph functions. In: Proc. International Conference on Machine Learning (ICML) (2014)
39. Vaswani, S.: Structured bandits and applications?: exploiting problem structure for better decision-making under uncertainty. Ph.D. thesis, University of British Columbia (2018). https://doi.org/http://dx.doi.org/10.14288/1.0375850, https://open.library.ubc.ca/collections/ubctheses/24/items/1.0375850
40. Wai, H.T., Segarra, S., Ozdaglar, A.E., Scaglione, A., Jadbabaie, A.: Blind community detection from low-rank excitations of a graph filter. IEEE Transactions on signal processing **68**, 436–451 (2019)
41. Waradpande, V., Kudenko, D., Khosla, M.: Deep reinforcement learning with graph-based state representations. arXiv:2004.13965 (2020)
42. de Wiele, T.V., Warde-Farley, D., Mnih, A., Mnih, V.: Q-learning in enormous action spaces via amortized approximate maximization. arXiv:2001.08116 (2020)
43. Yang, K., Dong, X., Toni, L.: Laplacian-regularized graph bandits: Algorithms and theoretical analysis. In: in Proc. International Conference on Artificial Intelligence and Statistics (AISTATS) (2020)
44. Yang, K., Toni, L.: Graph-based recommendation system. In: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP) (2018)
45. Yuan, K., Ying, B., Zhao, X., Sayed, A.H.: Exact Diffusion for Distributed Optimization and Learning — Part I: Algorithm Development. ArXiv **abs/1702.05122** (Feb 2017)
46. Zhang, H., Feng, T., Yang, G.H., Liang, H.: Distributed cooperative optimal control for multiagent systems on directed graphs: An inverse optimal approach. IEEE Transactions on Cybernetics **45**(7), 1315–1326 (July 2015)