



HAL
open science

Bandits on graphs and structures

Michal Valko

► **To cite this version:**

Michal Valko. Bandits on graphs and structures. Machine Learning [stat.ML]. École normale supérieure de Cachan - ENS Cachan, 2016. tel-01359757

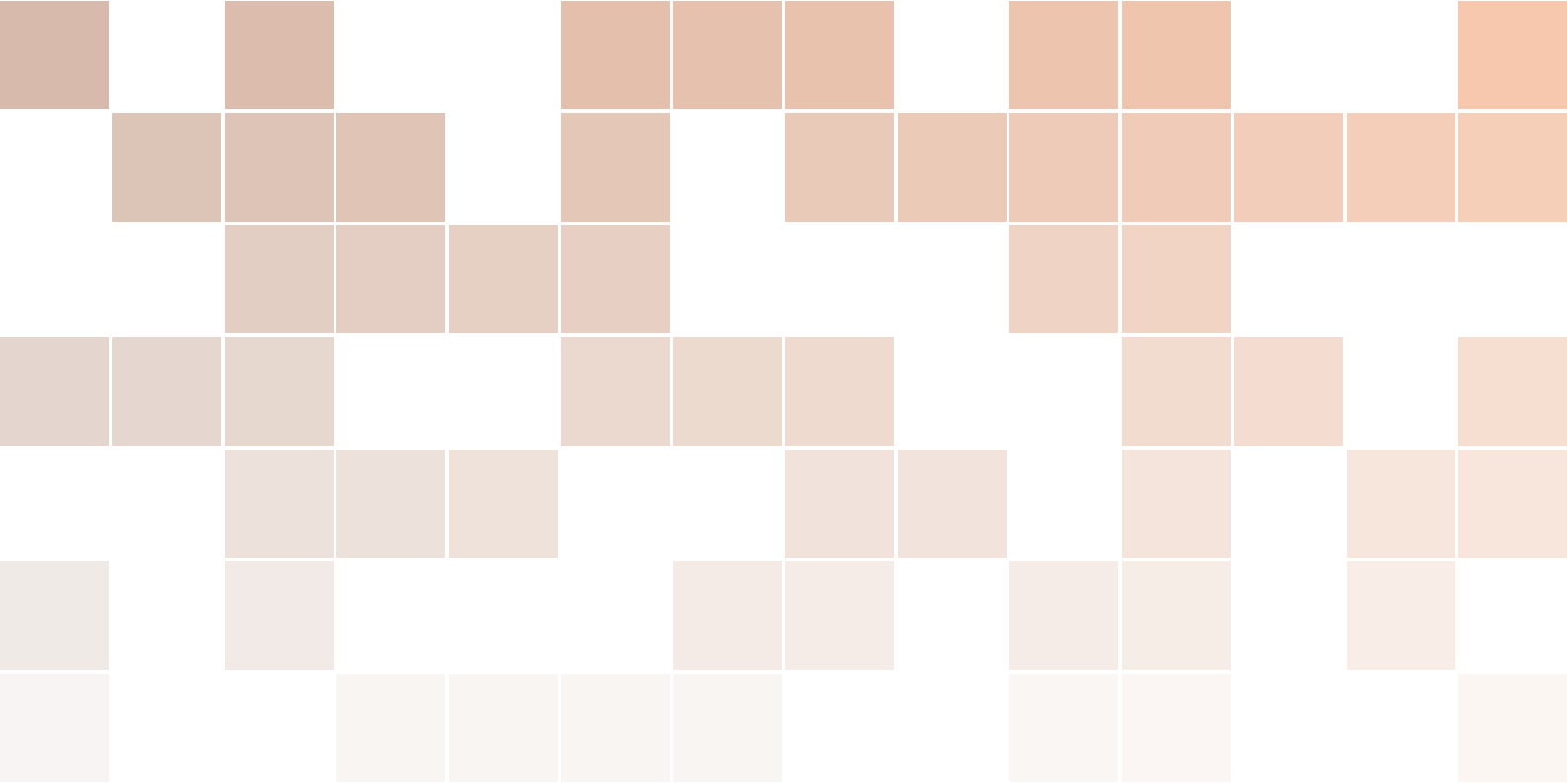
HAL Id: tel-01359757

<https://inria.hal.science/tel-01359757v1>

Submitted on 4 Sep 2016

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

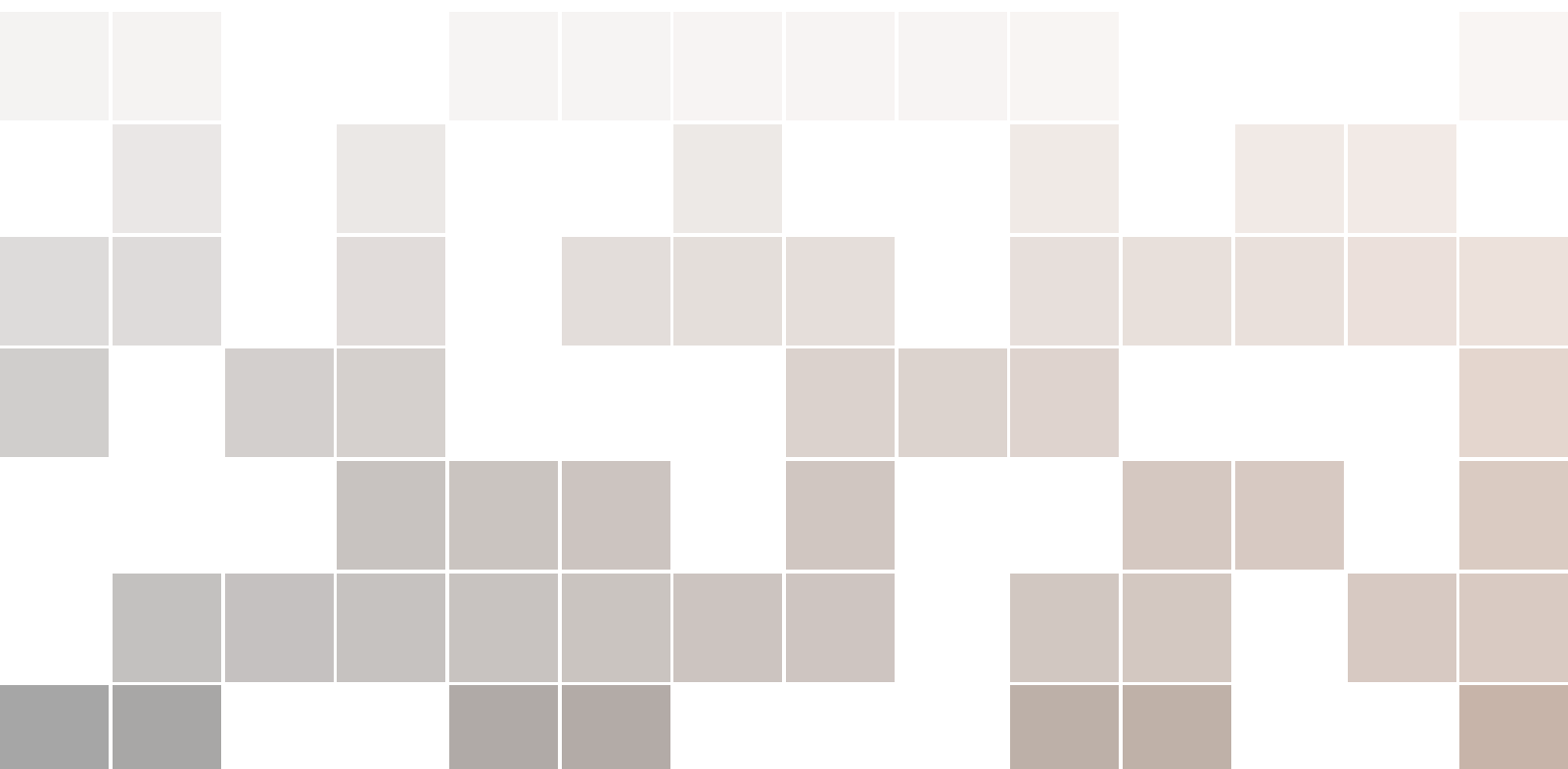
L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Bandits on Graphs and Structures

habilitation à diriger des recherches

Michal Valko



**HABILITATION À DIRIGER DES RECHERCHES DE
L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN**



Habilitation à Diriger des Recherches

Spécialité : **Mathématiques**

présentée par

Michal VALKO



Bandits on Graphs and Structures

Rapporteurs : M. Aurélien **GARIVIER** Université Toulouse
M. Gábor **LUGOSI** Universitat Pompeu Fabra
M. Vianney **PERCHET** ENSAE ParisTech

Soutenue le **15 Juin 2016** devant le jury composé de

M. Nicolas	VAYATIS	ENS de Cachan	Garant & Examineur
M. Aurélien	GARIVIER	Université Toulouse	Président & Rapporteur
M. Gábor	LUGOSI	Universitat Pompeu Fabra	Rapporteur
M. Vianney	PERCHET	ENSAE ParisTech	Rapporteur
M. Nicolò	CESA-BIANCHI	Università di Milano	Examineur
M. Mark	HERBSTER	University College London	Examineur
M. Rémi	MUNOS	DeepMind & Inria	Examineur

Contents

Summary	7
----------------------	----------

I	Graph bandits	
1	Smoothness of rewards	15
1.1	Spectral bandits	16
1.1.1	Effective dimension for spectral bandits	17
1.1.2	Algorithms for spectral bandits	17
1.1.3	Scalability and computational complexity	19
1.2	Related approaches to smoothness on graphs	20
1.2.1	Spectral bandits with different objectives	20
1.2.2	Smoothness of linear parameter vectors	20
1.2.3	Clusters of linear bandits, unimodal bandits, and reward from multiple nodes	21
1.3	Perspectives for graph-smooth rewards	21
1.3.1	Improvements for the effective dimension for spectral bandits	21
1.3.2	Applicability to recommender systems	22
2	Side observations	23
2.1	Undirected side observations	25
2.2	Directed side observations	26
2.2.1	Implicit exploration and Exp3-IX	27
2.2.2	Exp3.G	28
2.2.3	Combinatorial semi-bandit problems with side observations	28
2.3	Noisy side observations	30

2.4	Stochastic losses	33
2.4.1	Gaussian losses and side observations	33
2.5	Lower bounds and high-probability bounds	33
2.6	Perspectives for side observations	34
2.6.1	Beyond bandits	34
2.6.2	Graph generators	35
3	Influence maximization	39
3.1	Local influence and revelation bandits	40
3.2	Perspectives of bandit influence maximization	43
3.2.1	Global models of influence	43
3.2.2	Crawling bandits	44

II Stochastic bandits in large structured domains

4	Kernel bandits	49
4.1	Kernelized UCB	49
4.2	Analysis of KernelUCB	51
4.3	Relationship with GP-UCB	52
4.4	Perspectives of bandits for stochastic processes	53
5	Polymatroid bandits	55
5.1	Optimization on polymatroids	56
5.2	Combinatorial optimization on polymatroids	58
5.3	Learning model	58
5.4	The OPM algorithm	59
5.5	Discussion and perspectives of polymatroid bandits	60
6	Bandits for function optimization	63
6.1	Near-optimality dimension independent of a semi-metric	64
6.2	Hierarchical optimistic optimization: HOO	65
6.3	Unknown function smoothness: StoS00	66
6.4	Parallel optimistic optimization: P00	67
6.5	Applicability and perspectives of bandit function optimization	68
7	Infinitely many armed bandits	71
7.1	Learning setting	72
7.2	Lower bounds	73
7.3	SiRI and its upper bounds	74
7.4	Extensions of SiRI	76
	References	77

Many thanks to the committee ...



Aurélien Garivier
U Toulouse



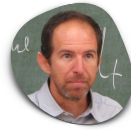
Gábor Lugosi
U Pompeu Fabra



Mark Herbster
U College London



Nicolas Vayatis
ENS Cachan



Nicolò Cesa-Bianchi
U Milano



Rémi Munos
Google DeepMind



Vianney Perchet
ENSAE ParisTech

... coauthors ...



Akram Erraqabi
U Montréal



Alessandro Lazaric
Sequel, Inria



Alexandra Carpentier
U Potsdam



Azin Ashkan
Technicolor Labs



Branislav Kveton
Adobe Research



Daniele Calandriello
Sequel, Inria



Gergely Neu
U Pompeu Fabra



Ilias Flaounas
Atlassian



Jean-Bastien Grill
Sequel, Inria



Julien Audiffren
CMLA, ENS Cachan



Manjesh Hanawal
Boston U



Mohammad Ghavamzadeh
Adobe Research



Nathan Korda
U of Oxford



Nello Cristianini
U Bristol



Odalric-Ambrym Maillard
Tao, Inria



Philippe Preux
Sequel, Inria



Rémi Munos
Google DeepMind



Shipra Agrawal
Columbia U



Tomáš Kocák
Sequel, Inria



Venkatesh Saligrama
Boston U



Yaakov Engel
Rafael



Zheng Wen
Adobe Research

... friends, colleagues, supporters, reviewers, students, future readers, and family.

Summary

The goal of this thesis is to investigate the *structural* properties of certain *sequential* problems in order to bring the solutions closer to a practical use. In the first part, we put a special emphasis on structures, that can be represented as *graphs* on actions, in the second part we study the large action spaces that can be of *exponential size* in the number of base actions or even *infinite*.

Graph bandits

Bandit problems are online decision-making problems where the only feedback given to the learner is a (noisy) reward of the chosen decision. In early sequential decision-making research, we treated each of the decisions *independently*. While this is enough when the number of actions is very small, it becomes difficult (both theoretically and in practice) when the set of potential actions comprises larger sets, such as a set of movies or products in a recommender system. The minimax regret guarantees scale as $\Theta(\sqrt{NT})$, where N is the number of actions and T is the time horizon. If N happens to be large (such as the number of movies that can be in millions), these guarantees are weak. Luckily, the problems become easier if there is an *efficient information sharing* between the actions. For instance, we will study the benefits of homophily (similar actions give similar rewards) and side information. Our goal is to take advantage of these similarities in order to (provably) learn *faster*. With respect to the guarantees we aspire to give, we aim to replace N (number of actions = number of nodes in a graph) with some *graph-dependent quantity*, possibly smaller than N if the graph structure is helpful. This part aspires to be a *survey* of all work done in graph bandits.

Smoothness of rewards in graph bandits

Consider an example when the actions are nodes on a given graph, e.g., recommending a movie in a graph of movies with movie similarities on edges. Then in many realistic situations, the rewards (i.e., the ratings) are smooth on this graph. This smoothness in graph bandits is a structural property that we leveraged in *spectral bandits* (Valko et al. 2014) to deliver a UCB-style algorithm (Auer et al. 2002a; Burnetas and Katehakis 1996) whose performance does not scale with the number of nodes but rather with the *effective dimension*, a quantity related to the *number of relevant eigenvectors* of the associated graph Laplacian, which is often much smaller in practice. Later, we gave a

computationally faster algorithm based on ThompsonSampling (Kocák et al. 2014b). Furthermore, we extended the results to the cost-sensitive setting called *cheap bandits* (Hanawal et al. 2015), relevant for radar applications, where the reward is an average of several nodes.

Side observations in graph bandits

Another setting, where we can learn faster using a graph structure, is the case when we receive some *side information*, as formalized by Mannor and Shamir (2011). This setting is a *partial observability model*, capturing situations where the information conveyed to the learner is between the full information and the bandit feedback. In the simplest variant, we assume that in addition to its own loss, the learner also gets to observe losses of some other actions — neighbor nodes on a graph. For instance, if we optimize click-through rate in the newsfeed recommendation, the additional information comes from the fact that several news feeds can refer to the same content, giving us the opportunity to infer clickthroughs for a number of assignments. This additional feedback allows for new algorithms where the regret guarantees improve from $\mathcal{O}(\sqrt{NT})$ to $\mathcal{O}(\sqrt{\alpha T})$ where α is the *independence number* of the graph and can range from 1 (complete graphs) to N (empty graphs). With the new *implicit exploration* technique employed in the **Exp3-IX** algorithm (Kocák et al. 2014a), we were able to relax the assumption needed by Alon et al. (2013) which required the knowledge of the graph before the action was chosen. This relaxation was also important, because it removed the costly exploration phase previously needed, including a computation of the dominating set in each time step. It also removed the need for the doubling trick and the need to aggregate several algorithms to appropriately tune the learning rate. We further extended the setting to the *combinatorial* side observation case, when the learner can select a subset of the nodes (potentially constrained by combinatorial constraints). This can be relevant for example in learning of bipartite matching between the users and the recommendations. We provided the FPL version of implicit exploration for this setting, achieving the regret of $\mathcal{O}(m^{3/2}\sqrt{\alpha T})$, where m is the number of nonzero components. Finally, in some graphs, the side observations are perturbed by *noise*. This is the case in *sensor networks*, where the communication reliability is a decreasing function of the distance. For this case, we extended this setting to the noisy information case (Kocák et al. 2016b), delivering an algorithm with $\mathcal{O}(\sqrt{\alpha^* T})$, where α^* is the *effective independence number*.

Influence maximization in graph bandits

For most of the graph bandits approaches, the algorithms need to have access to at least some portions of the graph to properly update their loss estimates. This is however not always possible in practice. Therefore, we formalized revealing bandits (Carpentier and Valko 2016), for the *influence maximization* in the stochastic setting, where the only information that the learner receives is the set of the influenced nodes. The goal is to find the most influential node without knowing the probabilities of influence between the node pairs. For this setting, we delivered a minimax optimal algorithm, *bandit revelator* (**BARE**) and showed that it attains $\mathcal{O}(\sqrt{D_* T})$ cumulative regret, where D_* is the *detectable dimension*, small when the unknown underlying graph has a small number of very influential nodes. Our setting considers only a simple model of *local* influence, but opens a way to study influence maximization in a sequential setting for more global influence models known in *computational social sciences* (Kempe et al. 2015).

Stochastic bandits in large structured domains

Not all structured action spaces are graphs and in the second part we address bandit setting with *large* (exponential or infinite) action spaces with *stochastic* rewards. While in the case of graph bandits we could opt to ignore the graph and use known multi-arm bandit strategies whose regret is worse by scaling with number of nodes N ; when the number of actions is exponential or infinite, this is no longer even an option. Using the *structure* in the large-action setting is *essential*.

Kernel bandits

Linear (contextual) bandits allow for information sharing between the rewards through the unknown, but *fixed* vector of weights via linear combination of features. This allows the regret bounds of linear bandit algorithms (LinRel, Auer 2002; LinUCB, Lihong Li et al. 2010; OFUL, Abbasi-Yadkori et al. 2011; LinearTS, Agrawal and N. Goyal 2013; LinearEliminator, Valko et al. 2014) to scale not with the number of actions, but with the size of the vector of weights D , i.e., the dimension of the context. In *kernel bandits*, the rewards may be a smooth function of contexts as given by some similarity function, while the set of the decision can still be discrete (GP-UCB, Srinivas et al. 2010; KernelUCB, Valko et al. 2013b). However, when the reward is an arbitrary linear function of context in the related RKHS, the dimension of the fixed vector of weights expressing the linear combination can be infinite, and, therefore, the straightforward use of linear bandit analysis does not apply. We show, however, that the regret of KernelUCB can be bounded in terms of the *effective dimension*, measuring the decay of eigenvalues of the covariance matrix in kernel regression and show its link to the *maximal information gain*, appearing in the analysis of GP-UCB.

Polymatroid bandits

When the potential actions in sequential learning form a combinatorial set, a very useful and specific structure is a *polymatroid*, where the associated offline learning problem with a known model can be solved optimally by a simple Greedy algorithm. Since some interesting problems are instances of sequential optimization on a polymatroid (e.g., the recommendation of *diverse items*) we propose and analyze OPM (Kveton et al. 2016), optimistic polymatroid optimization algorithm suited for this setting with a regret bound scaling with the *rank of the polymatroid*.

Bandits for function optimization

A very general structure is an arbitrary function. If we are to optimize such function sequentially, we may do it by extending the ideas from the bandit theory. In this case, the decision set is continuous, and in the more challenging setting, the (reward) function may be only locally smooth around one of its optima, while the *smoothness is unknown*. We first provided an algorithm for an easy class of functions (StoS00, Valko et al. 2013a, Preux et al. 2014) and later for a much wider class of difficult-to-optimize functions (POO, Grill et al. 2015).

Infinitely many-arms bandits

Even in the *continuous* case when *no topology* on the decision set is given to the learner, the learner still can take advantage of the structure if there is a certain quantity of near-optimal decisions. This setting was formally defined by Berry et al. (1997) as *infinitely many-arms bandits*, when the learner can sample from an infinite pool of decisions. Our contribution is the SiRI algorithm (Carpentier and Valko 2015), that is nearly minimax optimal in the *simple regret* setting and potentially useful in the best feature selection when we face a large number of candidates. The regret guarantees depend on a β -parameter characterizing the distribution of near-optimal arms.

Notation

By default, we use T for the number of rounds and N for the number of actions, e.g., the number of nodes in a graph. For any $a \in \mathbb{N}^+$, $[a]$ stands for the set of first a positive natural numbers, $[a] \stackrel{\text{def}}{=} \{1, 2, 3, \dots, a\}$. The rest of the notation is introduced when it is used. We also highlight the graph- or problem-dependent quantities as $d, \alpha, \alpha^*, \chi, \text{mas}, r, D_*, L, K, \tilde{d}$, and β .



Graph bandits

1	Smoothness of rewards	15
1.1	Spectral bandits	
1.2	Related approaches to smoothness on graphs	
1.3	Perspectives for graph-smooth rewards	
2	Side observations	23
2.1	Undirected side observations	
2.2	Directed side observations	
2.3	Noisy side observations	
2.4	Stochastic losses	
2.5	Lower bounds and high-probability bounds	
2.6	Perspectives for side observations	
3	Influence maximization	39
3.1	Local influence and revelation bandits	
3.2	Perspectives of bandit influence maximization	

In this section, we consider the variations of the following setting. There is a (known or unknown) graph \mathcal{G} , with the node set $\mathcal{V} = \{v_1, \dots, v_N\}$ of N nodes and the edge set \mathcal{E} . Every round $t = 1, \dots, T$, the learner picks I_t -th node (decision, arm, or action) v_{I_t} . At the same time, the environment (possibly adversarial) independently picks a vector of losses $\ell_t \in [0, 1]^N$ and the learner suffers the loss ℓ_{t, I_t} . As usual in online settings (Cesa-Bianchi and Lugosi 2006), the performance is measured in terms of (total expected) regret, which is the difference between a total loss received and the total loss of the best single action chosen in hindsight,

$$R_T = \max_{i \in [N]} \mathbb{E} \left[\sum_{t=1}^T (\ell_{t, I_t} - \ell_{t, i}) \right] \quad (1)$$

where the expectation integrates over the random choices made by the learning algorithm. Equivalently, instead of losses ℓ_t , we can consider rewards \mathbf{r}_t . If the nodes have associated context vectors, we will denote them by $\mathbf{x}_1, \dots, \mathbf{x}_N$, with $\mathbf{x}_i \in \mathbb{R}^D$. The goal of the learner is to minimize R_T .

General lower bound

Every setting in this part can be potentially treated with known algorithms and analyses for multi-arm bandits by *ignoring the presence of the graph* and considering the nodes as independent decisions. Ignoring the graph structure and using the known algorithms for either the stochastic or adversarial bandits is limited by the following lower bound.

Theorem 0.0.1 — Multi-arm bandit lower bound by Auer et al. (2002b). For any number of actions $N \geq 2$, there exists a distribution of losses (rewards) such that

$$\mathbb{E}[R_T] \geq \frac{1}{20} \min \left\{ \sqrt{NT}, T \right\}.$$

This part addresses the situations with a large number of nodes (actions, decisions, arms) N , which makes the regret scale unfavorably with N . In the rest of this chapter we describe the approaches that leverage the graph structure for specific settings with the regret bounds that essentially replace N in the lower bound of Theorem 0.0.1 with a possibly much smaller, *graph-dependent* quantity.

In the following three chapters, we consider three groups of graph bandit setups. For each of them, we aim to find efficient regret-minimization algorithms and find out an appropriate problem dependent quantity:

- **Smoothness of rewards** on a given graph is a setup where we exploit situations where neighboring nodes give similar rewards. In a specific, *spectral bandit* setting, we show that the relevant quantity is the *number of relevant eigenvectors* of the Laplacian of \mathcal{G} .
- **Side observations** extend the bandit feedback to case where we receive *additional* reward information from the nodes adjacent to v_{I_t} . We show that several variations of this setup are linked to the *independence number* of \mathcal{G} .
- **Influence maximization** is a problem with a specific reward structure that is a function of the whole graph. In this setting, we aim at finding the nodes with the maximum influence on the rest of the nodes. For a simple model of *local influence*, we define a new *detectable dimension* of the problem.

1. Smoothness of rewards

A *smooth graph function* is a function on a graph that returns similar values on neighboring nodes. This concept arises frequently in manifold and semi-supervised learning (X. Zhu 2008), and reflects the fact that the outcomes on the neighboring nodes tend to be similar. It is well-known (Belkin et al. 2004; Belkin et al. 2006) that a smooth graph function can be expressed as a linear combination of the eigenvectors of the graph Laplacian with smallest eigenvalues. Therefore, the problem of learning such a function can be cast as a regression problem on these eigenvectors. We will bring this concept to bandits as a special case of the (stochastic) setting defined in the beginning of this part. In particular, we study a bandit problem where the arms are the nodes of a graph and the expected payoff of pulling an arm is a smooth function on this graph.

One application is *targeted advertisement* in social networks. Here, the graph is a social network and our goal is to discover a part of the network that is interested in a given product. Interests of people in a social network tend to change smoothly (McPherson et al. 2001), because friends tend to have similar preferences. Therefore, we take advantage of this structure and formulate this problem as learning a smooth preference function on a graph.

Another application of our work is *recommender systems* (Jannach et al. 2010). In content-based recommendation (Chau et al. 2011), the user is recommended items that are similar to the items that the user rated highly in the past. The assumption is that users prefer similar items similarly. The similarity of the items can be measured for instance by a nearest neighbor graph (Billsus et al. 2000), where each item is a node and its neighbors are the most similar items (Figure 1.1).

In both applications described above, the learner (advertiser) has rarely the budget (time T) to try all the options even once. Furthermore, imagine that the learner is a movie recommender system

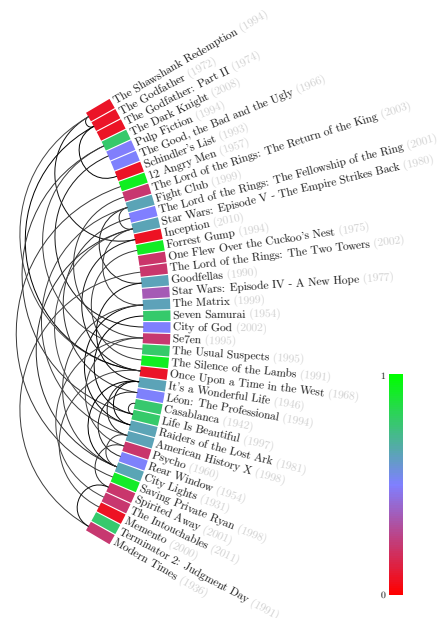


Figure 1.1: User preference for movies.

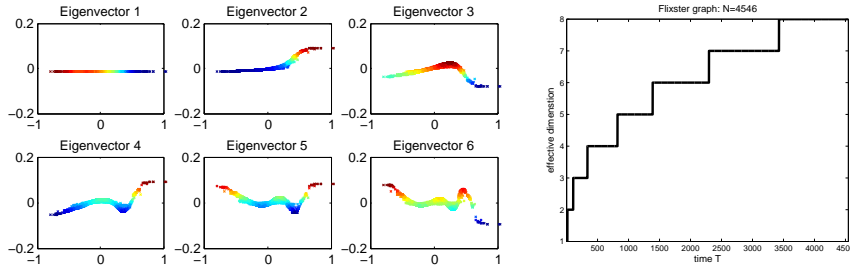


Figure 1.2: **Left:** Eigenvectors from the Flixster data corresponding to the smallest few eigenvalues projected onto the first principal component. Colors indicate the values. **Right:** Effective dimension as a function of time.

and would ask the user to rate all the movies before it starts producing relevant recommendations. Such a recommender system would be of little value. Yet, many bandit algorithms start with pulling each arm once. This is something that we cannot afford here and therefore, contrary to standard bandits, we consider the case $T \ll N$.

1.1 Spectral bandits

If the smooth graph function can be expressed as a linear combination of k eigenvectors of the graph Laplacian, and k is small and known, our learning problem can be solved using ordinary linear¹ bandits (Agrawal and N. Goyal 2013; Auer 2002; Lihong Li et al. 2010). In practice, k is problem specific and unknown. Moreover, the number of features k may approach the number of nodes N . Therefore, proper regularization is necessary, so that the regret of the learning algorithm does not scale with N . We are interested in the setting where the regret is independent of N and, therefore, this problem is nontrivial.

There are several ways to define the *smoothness* of the function f with respect to the graph G . We are using the one that is standard in spectral clustering and semi-supervised learning, defined as

$$S_G(f) = \frac{1}{2} \sum_{i,j \in [N]} w_{i,j} (f(i) - f(j))^2 = \mathbf{f}^\top \mathcal{L} \mathbf{f} = \mathbf{f}^\top \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^\top \mathbf{f} = \|\boldsymbol{\alpha}\|_{\mathbf{\Lambda}}^2 = \sum_{i=1}^N \lambda_i \alpha_i^2, \quad (1.1)$$

where $\mathbf{f} = (f(1), \dots, f(N))^\top$ is the vector of the function values, $\mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^\top$ is the eigendecomposition of graph laplacian \mathcal{L} where $\mathbf{\Lambda}$ is the diagonal matrix with entries $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_N$, and $\boldsymbol{\alpha} = \mathbf{Q}^\top \mathbf{f}$ is the representation of vector \mathbf{f} in the eigenbasis. The assumption on smoothness of the reward function with respect to the underlying graph is reflected in our belief that the value of the $S_G(f)$ is small and therefore, components of $\boldsymbol{\alpha}$ corresponding to the large eigenvalues should be small as well. If $\boldsymbol{\alpha}^*$ it the true (unknown) parameter vector, then in the stochastic setting, the reward of the chosen node v_t is assumed to be

$$r_t = \langle \mathbf{x}_{v_t}, \boldsymbol{\alpha}^* \rangle + \varepsilon_t,$$

where the noise ε_t is be R -sub-Gaussian for any t . In our setting, we have $\mathbf{x}_v \in \mathbb{R}^D$ and $\|\mathbf{x}_v\|_2 \leq 1$ for all \mathbf{x}_v . The goal of the recommender is to minimize the cumulative regret with respect to the strategy that always picks the best nodes w.r.t. $\boldsymbol{\alpha}^*$ and the definition of cumulative (pseudo) regret (1) simplifies to

$$R_T = T \max_{v \in \mathcal{V}} f_{\boldsymbol{\alpha}^*}(v) - \sum_{t=1}^T f_{\boldsymbol{\alpha}^*}(v_t), \text{ where } f_{\boldsymbol{\alpha}^*}(v) = \langle \mathbf{x}_v, \boldsymbol{\alpha}^* \rangle.$$

¹spectral bandits are therefore a special subcase of kernel bandits (Srinivas et al. 2010; Valko et al. 2013b)

1.1.1 Effective dimension for spectral bandits

The main benefit of expressing the reward vector in the spectral basis of the graph Laplacian (1.1) is that when only $d \ll N$ eigenvectors are enough to express the reward function well, then we can learn faster than with linear bandits, where the regret scales with D that denotes the ambient dimension, which is equal to N in the spectral setting.

In general, we assume a set of K vectors $\mathbf{x}_1, \dots, \mathbf{x}_K \in \mathbb{R}^N$ such that $\|\mathbf{x}_i\|_2 \leq 1$ for all i . For the spectral bandits, we have $K = N$. Moreover, since \mathbf{Q} is an orthonormal matrix, $\|\mathbf{x}_i\|_2 = 1$. Finally, since the first eigenvalue of a graph Laplacian is always zero, $\lambda_1^{\mathcal{L}} = 0$, we use $\mathbf{\Lambda} = \mathbf{\Lambda}_{\mathcal{L}} + \lambda \mathbf{I}$, with some positive regularizer λ , in order to have $\lambda_1 = \lambda > 0$. In order to present our algorithms and analyses, we introduce a notion of *effective dimension* d .

Definition 1.1.1 Let the **effective dimension** be the largest d such that:

$$(d-1)\lambda_d \leq \frac{T}{\log(1+T/\lambda)}$$

The effective dimension d (whose precise definition comes from our analysis) is small when the coefficients λ_i grow rapidly above T . This is the case when the dimension of the space D (and K) is much larger than T , such as in graphs from social networks with very large number of nodes N . In contrast, when the coefficients are all small, then d may be of the order of T , which would make the regret bounds useless. Figure 1.2 (right) shows how d behaves compared to D on generated graphs and graphs built from real-world data.²

The dependence of the effective dimension on T comes from the fact, that d is related to the number of *nonnegligible* dimensions characterizing the space where the solution to the penalized least-squares lies, since this solution is constrained to an ellipsoid defined by the inverse of the eigenvalues. In fact, for a small T , the axes of the ellipsoid corresponding to the large eigenvalues of \mathcal{L} are negligible with respect to the overall regret. Therefore, when T tends to infinity, all directions matter, thus the solution can be anywhere in a (bounded) space of dimension $D = N$. On the contrary, for a smaller T , the ellipsoid possesses a smaller number of nonnegligible dimensions. Notice that it is natural that this effective dimension depends on T as we consider the setting $T < N$. If we wanted to avoid T in the definition of d , we could define it as well in terms of N by replacing T by N in Definition 1.1.1, but this would only loosen its value.

Lower bound

While the known lower bound for linear bandits is $\Omega(\sqrt{DT})$, we can show a similar lower bound for spectral bandits $\Omega(\sqrt{dT})$, featuring the effective dimension. The main idea is to construct a graph composed of d almost disconnected components (Figure 1.3) and then reduce the setting to d -arm bandits with $\Omega(\sqrt{dT})$ lower bound (Theorem 0.0.1).

1.1.2 Algorithms for spectral bandits

Having expressed the rewards as a linear combination of eigenvectors, we can directly modify LinUCB (Lihong Li et al. 2010) to use *spectral penalty* (1.1) for the regularized least-squares estimate $\hat{\boldsymbol{\alpha}}_t$

$$\hat{\boldsymbol{\alpha}}_t = \arg \min_{\mathbf{w} \in \mathbb{R}^N} \left(\sum_{s=1}^t [\mathbf{x}_{I_s}^T \mathbf{w} - r_{I_s}]^2 + \|\mathbf{w}\|_{\mathbf{\Lambda}}^2 \right).$$

This gives us **SpectralUCB** with the regret scaling as $\tilde{\mathcal{O}}(d\sqrt{T})$.

²We set $\mathbf{\Lambda}$ to $\mathbf{\Lambda}_{\mathcal{L}} + \lambda \mathbf{I}$ with $\lambda = 0.01$, where $\mathbf{\Lambda}_{\mathcal{L}}$ is the graph Laplacian of the respective graph.

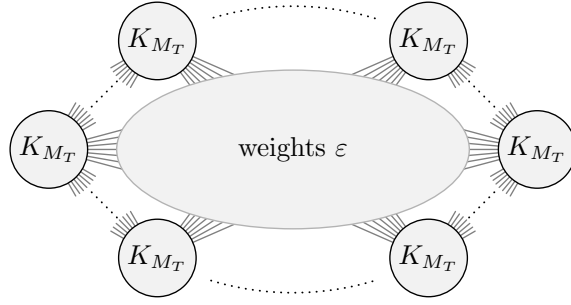


Figure 1.3: Weights within blocks K_{M_T} have value 1, otherwise ε . K_{M_T} is a complete graph on M_T nodes, with M_T being a function of T .

Theorem 1.1.1 — Regret of SpectralUCB by Valko et al. (2014). Let d be the effective dimension and λ be the minimum eigenvalue of Λ . If $\|\alpha\|_{\Lambda} \leq C$ and for all \mathbf{x}_a , $\mathbf{x}_a^{\top} \alpha \in [-1, 1]$, then the cumulative regret of SpectralUCB is with probability at least $1 - \delta$ bounded as

$$R_T \leq \left(4R\sqrt{d \log(1 + T/\lambda)} + 2\log(1/\delta) + 2C + 2\right) \sqrt{4dT \log(1 + T/\lambda)}.$$

- R** The constant C needs to be such that $\|\alpha\|_{\Lambda} \leq C$. If we set C too small, the true α will lie outside of the region and far from $\hat{\alpha}_t$, causing the algorithm to underperform. Alternatively, C can be time dependent, e.g., $C_t = \log T$. In such case, we do not need to know an upper bound on $\|\alpha\|_{\Lambda}$ in advance, but our regret bound would only hold after some t , when $C_t \geq \|\alpha\|_{\Lambda}$.

It is known that the available upper bound for LinUCB (Lihong Li et al. 2010), LinearTS (Agrawal and N. Goyal 2013) or OFUL (Abbasi-Yadkori et al. 2011) is not optimal for the linear bandit setting with finite number of arms in terms of dimension D . On the other hand, the algorithms SupLinRel or SupLinUCB achieve the optimal \sqrt{DT} regret. In the following, we likewise provide an algorithm that also scales better with d and achieves \sqrt{dT} regret. The algorithm is called SpectralEliminator (Valko et al. 2014) and works in phases, eliminating the arms that are not promising. The phases are defined by the time indexes $t_1 = 1 \leq t_2 \leq \dots$ and depend on some parameter β . The algorithm is in spirit similar to the ImprovedUCB by Auer and Ortner (2010). The main idea of SpectralEliminator is to divide the time steps into sets in order to introduce independence and allow the Azuma-Hoeffding inequality (Azuma 1967) to be applied. In the following theorem, we characterize the performance of SpectralEliminator and show that the upper bound on regret has \sqrt{d} improvement over SpectralUCB.

Theorem 1.1.2 — Regret of SpectralEliminator by Valko et al. (2014). Choose the phase starts as $t_j = 2^{j-1}$. Assume all rewards are in $[0, 1]$ and $\|\alpha\|_{\Lambda} \leq C$. For any $\delta > 0$, with probability at least $1 - \delta$, the cumulative regret of SpectralEliminator algorithm run with parameter $\beta = 2R\sqrt{14\log(2K(1 + \log_2 T)/\delta)} + C$ is bounded as:

$$R_T \leq 2 + 16 \left(2R\sqrt{14\log \frac{2K(1 + \log_2 T)}{\delta}} + C + \frac{1}{2} \right) \times \sqrt{dT \log_2(T) \log(1 + T/\lambda)}$$

- R** If we use $\mathbf{\Lambda} = \mathbf{I}$ in `SpectralEliminator`, we get a new algorithm, `LinearEliminator`, which is a competitor to `SupLinRel` (Auer 2002) or `SupLinUCB` (Chu et al. 2011) and as a corollary to Theorem 1.1.2 also enjoys $\tilde{\mathcal{O}}(\sqrt{DT})$ upper bound on the cumulative regret. Compared to `SupLinRel` or `SupLinUCB`, `LinearEliminator` and its analysis are much simpler.

1.1.3 Scalability and computational complexity

There are three main computational issues to address in order to make the proposed algorithms scalable: the computation of N UCBs, matrix inversion, and obtaining the eigenbasis which serves as an input to the algorithm. First, to speed up the computation of N UCBs in each time step, we use the lazy updates technique (Desautels et al. 2012) which maintains a sorted queue of UCBs and in practice leads to substantial speed gains. Second, to speed up matrix inversion we do iterative matrix inversion (Zhang 2005).

Finally, while the eigendecomposition of a general matrix is computationally difficult, Laplacians are symmetric diagonally dominant (SDD). This enables us to use fast SDD solvers such as CMG by (Koutis et al. 2011). Furthermore, using CMG we can find good approximations to the first L eigenvectors in $\mathcal{O}(Lm \log m)$ time, where m is the number of edges in the graph (e.g., $m = 10N$ in the Flixster experiment). CMG can easily work with N in millions. In general, we have $L = N$ but from our experience, a smooth reward function can be often approximated by dozens of eigenvectors. In fact, L can be considered as an upper bound on the number of eigenvectors we actually need. Furthermore, by choosing small L we not only reduce the complexity of eigendecomposition but also the complexity of the least-square problem being solved in each iteration.

Choosing a small L can significantly reduce the computation but it is important to choose L large enough so that still less than L eigenvectors are enough. This way, the problem that we solve is still relevant and our analysis applies. In short, the problem cannot be solved trivially by choosing first k relevant eigenvectors because k is unknown. Therefore, in practice, we choose the largest L such that our method is able to run.

Even with all those improvements, we may have to recompute the UCBs for many arms. As in linear bandits, `ThompsonSampling` (Thompson 1933) provides more computationally efficient alternative and we can easily derive a `ThompsonSampling` equivalent of `SpectralUCB`. This variant is called `SpectralTS` (Kocák et al. 2014b) and its upperbound also scales as $\tilde{\mathcal{O}}(d\sqrt{T})$.

Theorem 1.1.3 — Regret of `SpectralTS` by Kocák et al. (2014b). Let d be the effective dimension and λ be the minimum eigenvalue of $\mathbf{\Lambda}$. If $\|\boldsymbol{\alpha}\|_{\mathbf{\Lambda}} \leq C$ and for all \mathbf{x}_a , $\mathbf{x}_a^T \boldsymbol{\alpha} \in [-1, 1]$, then the cumulative regret of `SpectralTS` is with probability at least $1 - \delta$ bounded as

$$R_T \leq \frac{11g}{p} \sqrt{\frac{4+4\lambda}{\lambda} dT \log \frac{\lambda+T}{\lambda}} + \frac{1}{T} + \frac{g}{p} \left(\frac{11}{\sqrt{\lambda}} + 2 \right) \sqrt{2T \log \frac{2}{\delta}},$$

where $p = 1/(4e\sqrt{\pi})$ and

$$g = \sqrt{4 \log(TN)} \left(R \sqrt{6d \log \left(\frac{\lambda+T}{\delta\lambda} \right)} + C \right) + R \sqrt{2d \log \left(\frac{(\lambda+T)T^2}{\delta\lambda} \right)} + C.$$

- R** Substituting g and p , we see that the regret bound scales as $d\sqrt{T \log N}$. Note that $N = D$ could be exponential in d and we need to consider factor $\sqrt{\log N}$ in our bound. On the other hand, if N is indeed exponential in d , then our algorithm scales with $\log D \sqrt{T \log D} = \log(D)^{3/2} \sqrt{T}$ which is even better.

R Since ThompsonSampling is a Bayesian approach, it requires a prior to run and we choose it here to be a Gaussian. However, this does not pose any assumption whatsoever about the actual data both for the algorithm and the analysis. The only assumptions we make about the data are: (a) that the mean payoff is linear in the features, (b) that the noise is sub-Gaussian, and (c) that we know a bound on the Laplacian norm of the mean reward function. We provide a frequentist bound on the regret (and not an average over the prior) which is a much stronger worst case result.

1.2 Related approaches to smoothness on graphs

In this section, we review other graph bandit approaches that assume smoothness but either have a different objective or they assume smoothness in some other form.

1.2.1 Spectral bandits with different objectives

In the follow-up work on spectral bandits, there have been algorithms optimizing other objective function than the cumulative regret. First, in some sensor networks, sensing a node (pulling and arm) has an associated cost (Narang et al. 2013). In a particular, *cheap bandit* setting (Hanawal et al. 2015), it is cheaper to get an average of rewards of a set of nodes than a specific reward of a single one. For this setting, we proposed CheapUCB (Hanawal et al. 2015) that reduces the cost of sampling by 1/4 as compared to SpectralUCB, while maintaining $\tilde{O}(d\sqrt{T})$ cumulative regret. Next, Gu and Han (2014) study the online classification setting on graphs with bandit feedback, very similar to spectral bandits. The analysis of their algorithm delivers essentially the same bound on the regret, however, they need to know the number of relevant eigenvectors d . Moreover, Ma et al. (2015) consider several variants of Σ -optimality that favors specific exploration when selecting the nodes.

1.2.2 Smoothness of linear parameter vectors

Spectral bandit strategies are relevant to recommender systems but only consider a single user. However, the information sharing is possible and desirable also between the users. This is considered in *gang of bandits* (Cesa-Bianchi et al. 2013), a graph bandit setting where each node represents user i , and is a linear bandit itself with parameter \mathbf{w}_i , unknown to the learner. Each round, the learner gets a user index i_t (node) with a set of contexts C_t and has to chose a $\bar{\mathbf{x}}_t \in C_t$. The graph, in this case, represents a network of users and it is the parameters $\{\mathbf{w}_i\}_i$ that are assumed to be smooth on the given graph in a Laplacian way. The GOB.Lin algorithm (Cesa-Bianchi et al. 2013) exploits this smoothness and after each feedback and the local update of $\hat{\mathbf{w}}_i$, also brings $\{\hat{\mathbf{w}}_i\}_i$ closer together.

Theorem 1.2.1 — Regret of GOB.Lin by Cesa-Bianchi et al. (2013). With probability $1 - \delta$, the cumulative regret of GOB.Lin is upperbounded as

$$R_T \leq 4\sqrt{T \left(R \ln \frac{m_T}{\delta} + L(\mathbf{u}_1, \dots, \mathbf{u}_N) \right) \ln |m_T|}.$$

where R is the sub-Gaussianity of the noise, m_T is a $(DN) \times (DN)$ block-diagonal matrix with block being the covariance matrices of each node (linear bandit) and

$$L(\mathbf{u}_1, \dots, \mathbf{u}_N) = \sum_{i \in \mathcal{V}} \|\mathbf{u}_i\|^2 + \sum_{(i,j) \in \mathcal{E}} \|\mathbf{u}_i - \mathbf{u}_j\|^2,$$

The term $\sum_{(i,j) \in \mathcal{E}} \|\mathbf{u}_i - \mathbf{u}_j\|^2$ in the bound above reflects the smoothness of the reward vectors among the nodes (user) and can be thought of as the vector version of the smoothness constant C in spectral bandit bound (Theorem 1.1.1). The value of $\ln |m_T|$ can be of order $\tilde{O}(DN)$.

1.2.3 Clusters of linear bandits, unimodal bandits, and reward from multiple nodes

A slightly stronger assumption is considered by Gentile et al. (2014), where the nodes of the graph (users) can be clustered with respect to some unknown underlying clustering and the nodes within a cluster exhibit similar behavior. The regret bound of their CLUB algorithm scales roughly with the number of clusters instead of the number of nodes, but can be even better if there are big clusters with identical arms. S. Li et al. (2016) later extended the approach to *double clustering* where both the users and the items are assumed to appear in clusters (with the underlying clustering unknown to the learner) and Korda et al. (2016) consider a distributed extension.

Yet another assumption of a special graph reward structure is exploited by unimodal bandits (Combes and Proutière 2014; Yu and Mannor 2011). One of the settings considered by Yu and Mannor (2011) is a graph bandit setting where every path in the graph has unimodal rewards and therefore also imposes a specific kind of smoothness with respect to the graph topology.

In networked bandits (Fang and Tao 2014), the learner picks a node, but besides receiving the reward from that node, its reward is the sum of the rewards of the picked node and its neighborhood. The algorithm of Fang and Tao (2014), *NetBandits*, can also deal with changing topology, however, this has to be always revealed to the learner before it makes its decision.

1.3 Perspectives for graph-smooth rewards

We outline some future extensions of bandit learning on graphs with smooth rewards.

1.3.1 Improvements for the effective dimension for spectral bandits

While the effective dimension is related to the number of relevant eigenvectors, its precise definition (Definition 1.1.1) comes from the analysis of the regularized covariance matrix used in least-squares regression (Valko et al. 2014, Lemma 6). One possible improvement is to define the effective dimension as an earlier upper bound in the analysis, in particular, define it as

$$d_{\text{new}} = \frac{\max \log \prod_{i=1}^N \left(1 + \frac{t_i}{\lambda_i}\right)}{\log \left(1 + \frac{T}{\lambda}\right)}$$

where the max is taken over all possible non-negative integers $\{t_1, \dots, t_N\}$, such that $\sum_{i=1}^N t_i = T$. This improves the scaling of the regret bound with respect to d (Figure 1.4) and since we use d in the algorithm to avoid computation of the determinants, this new definition has also a practical impact.

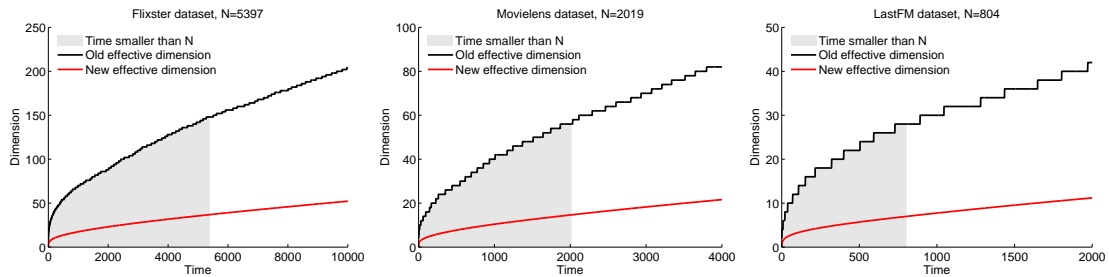


Figure 1.4: Difference between d_{new} and $2d$ for real world datasets. From left to right: Flixster dataset with $N = 5397$, MovieLens dataset with $N = 2019$, and LastFM dataset with $N = 804$.

Notice that d does not depend on the reward and while we have $\Omega(\sqrt{dT})$ lower bound in d , it is rather both effective dimension d and the upper bound on the reward smoothness C that together reflect the difficulty of spectral bandits. If the rewards are not smooth (C is large), then the problem is as difficult as learning N -arm bandits, no matter how small d is.³ Therefore, an interesting and more fundamental open question is the better understanding of the problem difficulty of spectral bandits with perhaps a *single* measure of difficulty.

1.3.2 Applicability to recommender systems

The oracle strategy for spectral bandits would always pick the most rewarding node. This is not desirable in many online recommender systems as this would mean watching the same movie or listening to the same song all the time. This is however not limiting, because, similar to linear bandits, the weight vector is the only thing that we are learning. This means that we can restrict the set of available arms to the ones that were not pulled yet and `SpectralUCB` or `SpectralTS` and their analyses extend to that situation.

On the other hand, unlike in linear bandits, changing the arm set in general (e.g., adding new arms) would require changes in the approach. The reason is that the spectral basis is assumed to be fixed (as is the standard basis for linear bandits) and is N -dimensional (there are N eigenvectors for N nodes). Adding a new arm (node) would often require the update of the basis and projecting the current estimate of the weight vector to this new basis.⁴

One aspect of spectral bandits is that it replaces the costly *feature-engineering* step needed for linear bandits with a pairwise similarity and this way circumvents feature selection. On the other hand, this approach is limited to a single user recommendation, unlike gang of bandits (Cesa-Bianchi et al. 2013), which however uses a linear bandit for each user needing feature construction. A useful future work would be the exploitation of smoothness in *both* the item and the user space.

A standard approach to smoothness or similarity of the rewards in the recommender systems is based on *low-rank matrix factorization* of the user-item matrix. Although there are already first results, studying this approach in bandit setting both for matrix factorization (Guillou et al. 2015; Guillou et al. 2016; Mary et al. 2015) and probabilistic matrix factorization (Kawale et al. 2015; Prasadnikov 2014; Tu and J.-L. Zhu 2015) based either on UCB or ThompsonSampling, they are mostly empirical. One of the difficulties is the nonconvexity of the non-negative matrix factorization and another one is the interplay between the rows and columns in the user-item matrix. Besides a more theoretical understanding of this approach, it would be interesting to relate it to spectral bandits and understand the tradeoffs coming from spectral smoothness vs. low-rank assumptions.

³This does not violate our upper bounds that obviously depend on C .

⁴Note that elimination algorithms, such as `SpectralEliminator` achieving $\tilde{O}(\sqrt{dT})$ regret, do not extend easily to the changing sets of arms.

2. Side observations

How to take advantage of richer feedback than a bandit one? In some situations, we can freely access or infer feedback for the actions that the online learner did not take. In *recommender systems*, this can be inferring interest about similar items (Figure 2.1, left) or accessories (Figure 2.2, left).

Another motivation is an online interaction in *sensor networks*, with sensors distributed in the area and where each sensor collects some information about the environment and can communicate with nearby sensors to share this information. Therefore, when the learner asks for data from a particular sensor, it can access the information from other, geographically close sensors.

If we equate the actions (sensor, choices, arms) with the nodes of a graph and the side information with the edges (Figure 2.1, right; and Figure 2.2, right) then we can see this setting as graph bandits with side observations. This setting was formally defined by Mannor and Shamir (2011) as an intermediate feedback protocol between bandit feedback and full information (learning with experts). The graph in this setting represents the *observation system* of side observations.

Parameters:

set of arms $[N]$, number of rounds T .

For all $t = 1, 2, \dots, T$ repeat

1. The environment picks a loss function $\ell_t : [N] \rightarrow [0, 1]$ and a directed graph \mathcal{G}_t with edge weights in $[0, 1]$.
2. Based on its previous observations (and possibly some source of randomness), the learner picks an action $I_t \in [N]$.
3. The learner suffers loss ℓ_{t, I_t} .
4. The learner observes \mathcal{G}_t and the feedback

$$\ell_{t, j} \text{ for all } j \text{ for all } (I_t \rightarrow j) \in \mathcal{G}_t$$

Figure 2.3: The protocol of bandit learning with side observations.

Figure 2.3 shows the learning protocol that we consider in this chapter. The different approaches vary depending whether the losses are stochastic or adversarial, whether the graphs are fixed or can

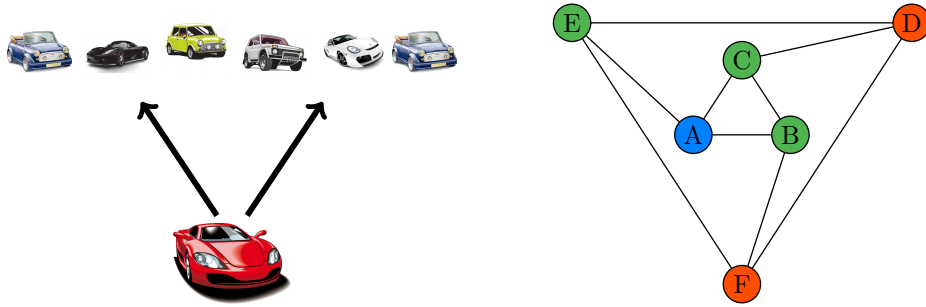


Figure 2.1: Side observations on **undirected** graphs. **Left:** *Recommendation example:* When a provider sees the interest in a particular sport car, they can assume the interest in other sport cars. **Right:** *Illustration of the feedback:* Whenever the learner asks for node A , it receives the feedback also for the nodes B , C , and E , but not for the rest.

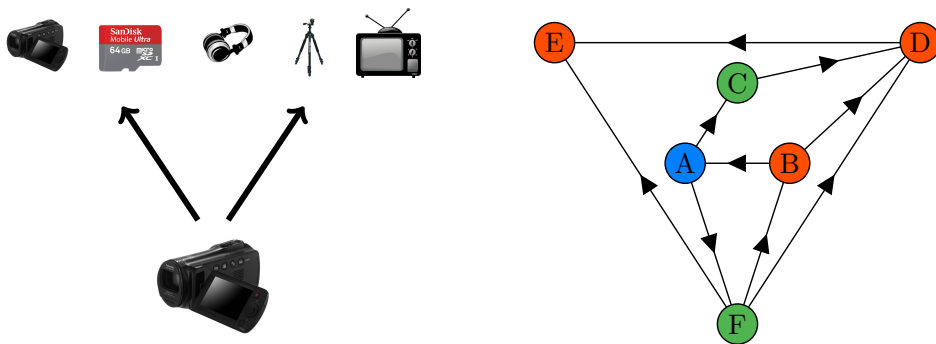


Figure 2.2: Side observations on **directed** graphs. **Left:** *Recommendation example:* When a provider sees the interest in video cameras, they can assume the interest in SD cards and tripods, but not necessarily vice versa. **Right:** *Illustration of the feedback:* Whenever the learner asks for node A , it receives the feedback also for the nodes C and F , but not for the rest.

change, whether they need to be revealed to the learner before it chooses the action or only after, and whether the graphs are directed or undirected. Table 2.1 lists the algorithms for the adversarial case with some of their properties that we later discuss in detail. Most of the algorithms for the

Algorithm	Reference	orientation	graph \mathcal{G}_t
ELP	Mannor and Shamir 2011	(un)directed	known before
Exp3-SET	Alon et al. 2013	undirected	only after
Exp3-DOM	Alon et al. 2013	(un)directed	known before
Exp3-IX, FPL-IX	Kocák et al. 2014a	(un)directed	only after
Exp3.G	Alon et al. 2015	(un)directed	only after

Table 2.1: Graph bandit algorithms for learning with side observations with nonstochastic losses

setting are graph variants of Exp3 (template shown in Algorithm 1) and vary by how they define their node sampling distribution (Line 7, Algorithm 1) and how they construct their loss estimates (Line 12, Algorithm 1). We will discuss these choices in the rest of the chapter. Before that, we define two graph-dependent quantities that will be used to state the regret bounds.

Algorithm 1 Exp3 template for graph bandits with side observations

-
- 1: **Input:** Set of actions $\mathcal{S} = [N]$, parameters $\gamma \in (0, 1)$, $\eta_t > 0$ for $t \in [T]$.
 - 2: **for** $t = 1$ **to** T **do**
 - 3: $w_{t,i} \leftarrow (1/N) \exp(-\eta_t \widehat{L}_{t-1,i})$ for $i \in [N]$
 - 4: An adversary privately chooses losses $\ell_{t,i}$ for $i \in [N]$ and generates a graph \mathcal{G}_t
 - 5: **Necessary for some algorithms:** Observe graph \mathcal{G}_t
 - 6: $W_t \leftarrow \sum_{i=1}^N w_{t,i}$
 - 7: **Define probabilities:** $p_{t,i}$, default: $p_{t,i} = \frac{w_{t,i}}{W_t}$
 - 8: Choose $I_t \sim \mathbf{p}_t = (p_{t,1}, \dots, p_{t,N})$
 - 9: Observe graph \mathcal{G}_t
 - 10: Observe pairs $\{i, \ell_{t,i}\}$ for $(I_t \rightarrow i) \in \mathcal{G}_t$
 - 11: $o_{t,i} \leftarrow \sum_{(j \rightarrow i) \in \mathcal{G}_t} p_{t,j}$ for $i \in [N]$
 - 12: **Define loss estimates:** $\widehat{\ell}_{t,i}$, default: $\widehat{\ell}_{t,i} \leftarrow \frac{\ell_{t,i}}{o_{t,i}} \mathbb{1}_{\{(I_t \rightarrow i) \in \mathcal{G}_t\}}$ for $i \in [N]$
 - 13: $\widehat{L}_{t,i} \leftarrow \widehat{L}_{t-1,i} + \widehat{\ell}_{t,i}$ for $i \in [N]$
 - 14: **end for**
-

Definition 2.0.1 The **independence set** of graph \mathcal{G}_t is a set of nodes, for which no pair is adjacent. The maximum possible size of such set is called **independence number** α_t .

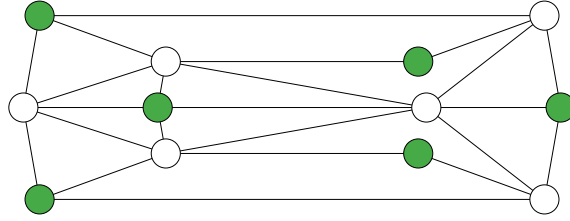


Figure 2.4: *Example:* Independence set of size 6.

Definition 2.0.2 The **clique-partition number** χ_t of graph \mathcal{G}_t is the smallest number of cliques that partition all the nodes.

- R** If α_t and χ_t are the independence and the clique-partition numbers of the same graph, then $\alpha_t \leq \chi_t$, since any clique can have at most one node from the independence set.

2.1 Undirected side observations

In this section, we consider undirected observations from Figure 2.1, which means that in every round, graph \mathcal{G}_t is undirected (symmetric). The first algorithm, ELP (Mannor and Shamir 2011), uses unbiased loss estimates (default setting in Line 12, Algorithm 1). Moreover, ELP's sampling probability distribution over the nodes (Line 7, Algorithm 1) is

$$p_{t,i} = (1 - \gamma) \frac{w_{t,i}}{\sum_{j=1}^N w_{t,j}} + \gamma s_{t,i}, \text{ where } \{s_{t,i}\}_{i \in [N]} = \arg \max_{\forall i s_{t,i} \geq 0, \sum_i s_{t,i} = 1} \min_{j \in [N] (j \rightarrow i) \in \mathcal{G}_t} \sum_{l \in \mathcal{G}_t} s_{t,l},$$

where $\{s_{t,i}\}_{i \in [N]}$ can be found using linear programming (LP), given the graph is revealed by the environment in Line 5 (Algorithm 1). Intuitively, ELP *mixes* in a distribution that is not uniform but is aware of the observation system (graph) and thus aims at distributing the exploration equally. Furthermore, ELP uses gains instead of losses.

Theorem 2.1.1 — Regret of ELP by Mannor and Shamir (2011). Setting the learning rate $\eta_t = \sqrt{(\log N) / (3 \sum_{t=1}^T \alpha_t)}$ and mixing rate $\gamma_t = \eta_t / (\min_{j \in [N]} \sum_{(j \rightarrow l) \in \mathcal{G}_t} s_{t,l})$, the expected regret of ELP is upper bounded as

$$R_T \leq \sqrt{3 \left(\sum_{t=1}^T \alpha_t \right) \log N}.$$

ELP needs to see \mathcal{G}_t revealed before the action is taken, to run the LP to tune its learning rate. This was fixed later by Exp3-SET (Alon et al. 2013), which does not need either. Exp3-SET uses losses instead of rewards, and differs from Exp3 only by the loss estimates (same as for ELP). For the sampling distribution, it uses simple Exp3 weighting without mixing,

$$p_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^N w_{t,j}},$$

and thus does not need to know the graph in Line 5 (Algorithm 1) to provide essentially the same guarantees on the regret.

Theorem 2.1.2 — Regret of Exp3-SET by Alon et al. (2013). Setting the learning rate $\eta_t = \sqrt{(2 \log N) / (\sum_{t=1}^T \alpha_t)}$, the expected regret of Exp3-SET is upper bounded as

$$R_T \leq \sqrt{2 \left(\sum_{t=1}^T \alpha_t \right) \log N}.$$

Knowledge of \mathcal{G}_t

Note that both ELP and Exp3-SET, set their learning parameter η_t as a function of $\sum_{t=1}^T \alpha_t$. This can be however avoided for both, by running an additional Exp3 algorithm on top it, at the price of an additional log factor (Mannor and Shamir 2011). Therefore, Exp3-SET can avoid any knowledge of \mathcal{G}_t before it picks a node. However, note that both of them need some knowledge of the graph *after* the node I_t is picked. In particular, they need it to construct the loss estimates in Line 12 of Algorithm 1. Since the algorithms update not only the loss estimate of I_t -th node but also of its neighbors in \mathcal{G}_t , the algorithms require also the knowledge of the neighbors of neighbors of I_t , the *second neighborhood* of I_t . This is shared by many algorithms in this chapter. While the knowledge of the first neighborhood is a very reasonable assumption (we know from which nodes the observations came from), the knowledge of the second neighborhood may not be always available in practical deployments.

2.2 Directed side observations

We now turn our attention to directed graphs from Figure 2.2. ELP from Section 2.1 can be used without modification, but the upper bound given by Mannor and Shamir (2011) only gives the version of Theorem 2.1.1, with clique-partition number χ_t instead of independence number α_t . Exp3-SET can be also used, however Alon et al. (2013), show the graph and the distribution of the sampling probabilities for which the key quantity (coming from Line 12, Algorithm 1) cannot be upper bounded by the independence number α . Therefore Alon et al. (2013) designed Exp3-DOM whose guarantees were proved to be functions of the *independence number*, which gives either equal or a better guarantee on the regret. To control the problematic quantity (discussed later), Exp3-DOM controls the loss estimates $\hat{\ell}_{t,i}$ by mixing in a uniform distribution in Line 7 of Algorithm 1, supported on the dominating set of the directed graph \mathcal{G}_t (set of the nodes that have the directed edges to the rest of the graph). This achieves the desired bound but comes with a few disadvantages. First, \mathcal{G}_t has to be revealed to the learner at the beginning of each round and so we

get the same limitation as for ELP. Second, depending on the size of the dominating set, Exp3-DOM needs to run $\log N$ instances to properly set the node sampling distribution. Finally, since the rounds where to use each instance are random, Exp3-DOM needs to use the doubling trick to optimally set γ_t and η_t .

2.2.1 Implicit exploration and Exp3-IX

Exp3-DOM of Alon et al. (2013) needed to know \mathcal{G}_t before choosing the action, to control the loss estimates. In this section we show how to achieve a similar behavior *without* the knowledge of \mathcal{G}_t (Kocák et al. 2014a). In particular, we propose the simplest exploration scheme imaginable, which consists of *merely pretending to explore*. Precisely, we simply sample our action I_t from the distribution defined as the default setting without explicitly mixing with any exploration distribution. Let $\mathcal{F}_{t-1} = \sigma(I_{-1}, \dots, I_1)$ capture the interaction history up to time t . Our key trick is to define the loss estimates for all arms i as

$$\widehat{\ell}_{t,i} = \frac{\ell_{t,i}}{o_{t,i} + \gamma_t} \mathbb{1}_{\{(I_t \rightarrow i) \in \mathcal{G}_t\}}, \quad \text{where } o_{t,i} = \mathbb{E}[O_{t,i} | \mathcal{F}_{t-1}] \stackrel{\text{def}}{=} \mathbb{P}[(I_t \rightarrow i) \in \mathcal{G}_t | \mathcal{F}_{t-1}]$$

and $\gamma_t > 0$ is a parameter of our algorithm. It is easy to check that $\widehat{\ell}_{t,i}$ is a *biased* estimate of $\ell_{t,i}$. The nature of this bias, however, is very special. First, observe that $\widehat{\ell}_{t,i}$ is an *optimistic* estimate of $\ell_{t,i}$ in the sense that $\mathbb{E}[\widehat{\ell}_{t,i} | \mathcal{F}_{t-1}] \leq \ell_{t,i}$. That is, our bias always ensures that, on expectation, we underestimate the loss of any fixed arm i . Even more importantly, our loss estimates also satisfy

$$\begin{aligned} \mathbb{E} \left[\sum_{i=1}^N p_{t,i} \widehat{\ell}_{t,i} \middle| \mathcal{F}_{t-1} \right] &= \sum_{i=1}^N p_{t,i} \ell_{t,i} + \sum_{i=1}^N p_{t,i} \ell_{t,i} \left(\frac{o_{t,i}}{o_{t,i} + \gamma_t} - 1 \right) \\ &= \sum_{i=1}^N p_{t,i} \ell_{t,i} - \gamma_t \sum_{i=1}^N \frac{p_{t,i} \ell_{t,i}}{o_{t,i} + \gamma_t}, \end{aligned} \tag{2.1}$$

that is, the bias of the estimated losses *suffered by our algorithm* is directly controlled by γ_t . As we will see in the analysis, it is sufficient to control the bias of our own estimated performance as long as we can guarantee that the loss estimates associated with any fixed arm are optimistic—which is precisely what we have. Note that this slight modification ensures that the denominator of $\widehat{\ell}_{t,i}$ is lower bounded by $p_{t,i} + \gamma_t$, which is a very similar property as the one achieved by the exploration scheme used by Exp3-DOM. We call the above loss estimation method *implicit exploration* or IX, as it gives rise to the same effect as explicit exploration without actually having to implement any exploration policy. In fact, explicit and implicit explorations can both be regarded as two different approaches for bias-variance tradeoff: while explicit exploration biases the *sampling distribution* of I_t to reduce the variance of the loss estimates, implicit exploration achieves the same result by biasing *the loss estimates themselves*.

From this point on, we take a somewhat more predictable course and define our algorithm **Exp3-IX** as a variant of Exp3 using the IX loss estimates. One of the twists is that **Exp3-IX** is actually based on the adaptive-learning-rate variant of Exp3 by Auer et al. (2002b), which avoids the necessity of prior knowledge of the observability graphs in order to set a proper learning rate. This algorithm is defined by setting $\widehat{L}_{t-1,i} = \sum_{s=1}^{t-1} \widehat{\ell}_{s,i}$ and for all $i \in [N]$ computing the weights as

$$w_{t,i} = (1/N) e^{-\eta_t \widehat{L}_{t-1,i}}.$$

These weights are then used to construct the sampling distribution of I_t as defined in Line 7 of Algorithm 1. As a result **Exp3-IX** does not even need to know the number of rounds T and our regret bound scales with the *average* independence number $\bar{\alpha}$ of the graphs played by the adversary rather than the largest of these numbers. **Exp3-IX** employs adaptive learning rate and unlike

Exp3-DOM, it does not need to use a doubling trick to be anytime or to aggregate outputs of multiple algorithms to optimally set their learning rates. The upper bound on the regret is stated below.

Theorem 2.2.1 — Regret of Exp3-IX by Kocák et al. (2014a). The regret of Exp3-IX satisfies

$$R_T \leq 4\sqrt{(N + 2\sum_{t=1}^T (H_t \alpha_t + 1)) \log N},$$

where

$$H_t = \log \left(1 + \frac{[N^2 \sqrt{tN/\log N}] + N}{\alpha_t} \right) = \mathcal{O}(\log(NT)).$$

2.2.2 Exp3.G

Learning on graphs with directed side observations is the special setting of a more general graph feedback related to partial monitoring, for which Alon et al. (2015) proposed the Exp3.G algorithm. Exp3.G also follows the template of Algorithm 1 and mixes in a uniform distribution over the nodes for the sampling distribution,

$$p_{t,i} = (1 - \gamma) \frac{w_{t,i}}{\sum_{j=1}^N w_{t,j}} + \frac{\gamma}{N},$$

which means that it does not need to know the graph \mathcal{G}_t for this step. The analysis of Exp3.G differs from the typical analysis of Exp3-style algorithms by using an improved second-order regret bound that considers separately small and large losses for a better control of variance. Exp3.G with proper tuning also achieves $\mathcal{O}(\sqrt{\alpha T \log(NT)})$ regret bound and can be generalized to the case when the graph is changing and when α (used for parameter tuning) is unknown using either the doubling trick or an adaptive learning rate employed by Exp3-IX. Exp3.G and its analysis can be however used in the more general feedback settings discussed in Section 2.6.1.

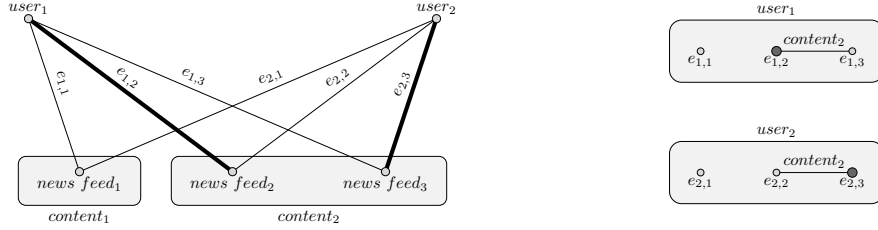
2.2.3 Combinatorial semi-bandit problems with side observations

We now turn our attention to the setting of online combinatorial optimization (see Audibert et al. 2014; Cesa-Bianchi and Lugosi 2012; Koolen et al. 2010). In this variant of the online learning problem, the learner has access to a possibly huge action set $\mathcal{S} \subseteq \{0, 1\}^N$ where each action is represented by a binary vector \mathbf{v} of dimensionality N . In what follows, we assume that $\|\mathbf{v}\|_1 \leq m$ holds for all $\mathbf{v} \in \mathcal{S}$ and some $1 \leq m \ll N$, with the case $m = 1$ corresponding to the multi-armed bandit setting considered in the previous section. In each round $t = 1, 2, \dots, T$ of the decision process, the learner picks an action $\mathbf{V}_t \in \mathcal{S}$ and incurs a loss of $\mathbf{V}_t^\top \boldsymbol{\ell}_t$. At the end of the round, the learner receives some feedback based on its decision \mathbf{V}_t and the loss vector $\boldsymbol{\ell}_t$. The regret of the learner is defined as

$$R_T = \max_{\mathbf{v} \in \mathcal{S}} \mathbb{E} \left[\sum_{t=1}^T (\mathbf{V}_t - \mathbf{v})^\top \boldsymbol{\ell}_t \right].$$

In this section, we define a new feedback scheme situated between the semi-bandit and the full-information schemes. In particular, we assume that the learner gets to observe the losses of some other components not included in its own decision vector \mathbf{V}_t . Similarly to the model of Alon et al. (2013), the relation between the chosen action and the side observations are given by a directed observability \mathcal{G}_t . We refer to this feedback scheme as *semi-bandit with side observations*. As an example, consider the situation shown on Figure 2.5a. In this simple example, we want to suggest one out of three news feeds to each user, that is, we want to choose a matching on the

graph shown on Figure 1a which covers the users. Assume that news feeds 2 and 3 refer to the same content, so *whenever we assign news feed 2 or 3 to any of the users, we learn the value of both of these assignments*. The relations between these assignments can be described by a graph structure (shown on Figure 2.5b), where nodes represent user-news feed assignments, and edges mean that the corresponding assignments reveal the clickthroughs of each other. For a more compact representation, we can group the nodes by the users, and rephrase our task as having to choose one node from each group. Besides its own reward, each selected node reveals the rewards assigned to all their neighbors.



(a) The thick edges represent one potential matching of users to feeds, grouped news feeds show the same content. (b) Connected feeds mutually reveal each others clickthroughs.

Figure 2.5: Users and news-feeds example of complex actions with side observations.

Figure 2.6 shows what happens in general, as Figure 2.2 (right) does for simple actions.

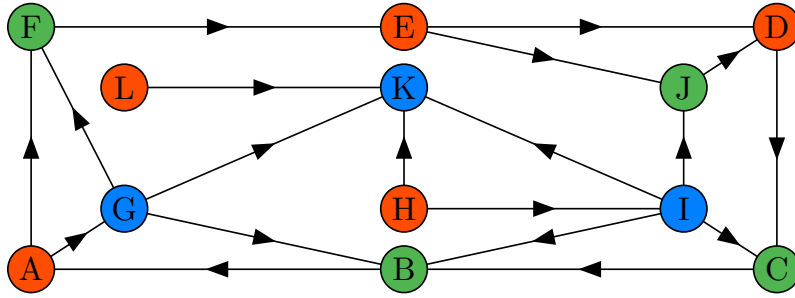


Figure 2.6: *Illustration of the feedback for complex actions*: Whenever the learner asks for nodes $G, I,$ and K , it receives the feedback also for the nodes $B, C, J,$ and F , but not for the rest.

While we could extend **Exp3-IX** to this setting, combinatorial **Exp3-IX** could rarely be implemented efficiently—we refer to Cesa-Bianchi and Lugosi (2012) and Koolen et al. (2010) for some positive examples. As one of the main concerns in this chapter is computational efficiency, we take a different approach: we propose a variant of FPL (Hannan 1957; Kalai and Vempala 2005) that efficiently implements the idea of implicit exploration in combinatorial semi-bandit problems with side observations. In each round t , FPL bases its decision on some estimate $\widehat{\mathbf{L}}_{t-1} = \sum_{s=1}^{t-1} \widehat{\ell}_s$ of the total losses $\mathbf{L}_{t-1} = \sum_{s=1}^{t-1} \ell_s$ as follows:

$$\mathbf{V}_t = \arg \min_{\mathbf{v} \in \mathcal{J}} \mathbf{v}^\top \left(\eta_t \widehat{\mathbf{L}}_{t-1} - \mathbf{Z}_t \right). \quad (2.2)$$

Here, $\eta_t > 0$ is a parameter of the algorithm and \mathbf{Z}_t is a perturbation vector with components drawn independently from an exponential distribution with unit expectation. The power of FPL lies in that it only requires an oracle that solves the (offline) optimization problem $\min_{\mathbf{v} \in \mathcal{J}} \mathbf{v}^\top \boldsymbol{\ell}$ and thus can be used to turn any efficient offline solver into an online optimization algorithm with strong

guarantees. To define our algorithm precisely, we need some further notation. We redefine \mathcal{F}_{t-1} to be $\sigma(\mathbf{V}_{t-1}, \dots, \mathbf{V}_1)$, $O_{t,i}$ to be the indicator of the observed *component* and let

$$q_{t,i} = \mathbb{E}[V_{t,i} | \mathcal{F}_{t-1}] \quad \text{and} \quad o_{t,i} = \mathbb{E}[O_{t,i} | \mathcal{F}_{t-1}].$$

The most crucial point of our algorithm is the construction of our loss estimates. To implement the idea of implicit exploration by optimistic biasing, we apply a modified version of the geometric resampling method of Neu and Bartók (2013) constructed as follows: Let $\mathbf{O}'_t(1), \mathbf{O}'_t(2), \dots$ be independent copies¹ of \mathbf{O}_t and let $U_{t,i}$ be geometrically distributed random variables for all $i \in [N]$ with parameter γ_t . We let

$$K_{t,i} = \min(\{k : O'_{t,i}(k) = 1\} \cup \{U_{t,i}\}) \quad (2.3)$$

and define our loss-estimate vector $\widehat{\ell}_t \in \mathbb{R}^N$ with its i -th element as

$$\widehat{\ell}_{t,i} = K_{t,i} O_{t,i} \ell_{t,i}. \quad (2.4)$$

By definition, we have $\mathbb{E}[K_{t,i} | \mathcal{F}_{t-1}] = 1/(o_{t,i} + (1 - o_{t,i})\gamma_t)$, implying that our loss estimates are *optimistic* in the sense that they lower bound the losses in expectation:

$$\mathbb{E}[\widehat{\ell}_{t,i} | \mathcal{F}_{t-1}] = \frac{o_{t,i}}{o_{t,i} + (1 - o_{t,i})\gamma_t} \ell_{t,i} \leq \ell_{t,i}.$$

Here we used the fact that $O_{t,i}$ is independent of $K_{t,i}$ and has expectation $o_{t,i}$ given \mathcal{F}_{t-1} . We call this algorithm Follow-the-Perturbed-Leader with Implicit eXploration (**FPL-IX**, Kocák et al. 2014a). Note that the geometric resampling procedure can be terminated as soon as $K_{t,i}$ becomes well-defined for all i with $O_{t,i} = 1$. As noted by Neu and Bartók (2013), this requires generating at most N copies of \mathbf{O}_t on expectation. As each of these copies requires one access to the linear optimization oracle over \mathcal{S} , we conclude that the expected running time of **FPL-IX** is at most N times that of the expected running time of the oracle. A high-probability guarantee of the running time can be obtained by observing that $U_{t,i} \leq \log(\frac{1}{\delta})/\gamma_t$ holds with probability at least $1 - \delta$ and thus we can stop sampling after at most $N \log(\frac{N}{\delta})/\gamma_t$ steps with probability at least $1 - \delta$. The regret guarantee for **FPL-IX** using the approximation $\widetilde{\alpha}_t$ of α_t is stated below.

Theorem 2.2.2 — Regret of FPL-IX by Kocák et al. (2014a). Assume that for all $t \in [T]$, $\alpha_t/C \leq \widetilde{\alpha}_t \leq \alpha_t \leq N$ for some $C > 1$. Setting $\eta_t = \gamma_t = \sqrt{(\log N + 1) / (m(N + \sum_{s=1}^{t-1} \widetilde{\alpha}_s))}$ and assuming $mN > 4$, the regret of **FPL-IX** satisfies

$$R_T \leq Hm^{3/2} \sqrt{(N + C \sum_{t=1}^T \alpha_t) (\log N + 1)}, \quad \text{where } H = \mathcal{O}(\log(mNT)).$$

2.3 Noisy side observations

Until now in this chapter, we studied situations when the learner observes losses associated with some additional actions besides its own loss. This setting fails to address one important practical concern: in reality, one can rarely expect *perfect* side-observations to be available. In the current section, we propose a similar model that can incorporate *imperfect* side-observations corrupted by various levels of noise, depending on the problem structure.

¹Such independent copies can be simply generated by sampling independent copies of \mathbf{V}_t using the FPL rule (2.2) and then computing $\mathbf{O}'_t(k)$ using the observability \mathcal{G}_t . Notice that this procedure requires no interaction between the learner and the environment, although each sample requires an oracle access.

Parameters:

set of arms $[N]$, number of rounds T .

For all $t = 1, 2, \dots, T$ repeat

1. The environment picks a loss function $\ell_t : [N] \rightarrow [0, 1]$ and a directed **weighted** graph \mathcal{G}_t with edge weights in $[0, 1]$.
2. Based on its previous observations (and possibly some source of randomness), the learner picks an action $I_t \in [N]$.
3. The learner suffers loss ℓ_{t, I_t} .
4. The learner observes \mathcal{G}_t and the feedback

$$c_{t,i} = s_{t,(I_t,i)} \cdot \ell_{t,i} + (1 - s_{t,(I_t,i)}) \cdot \xi_{t,i}$$

for every arm $i \in [N]$.

Figure 2.8: The protocol of online learning with **noisy** observations.

As an illustration of noisy setting, consider the problem of controlling solar panels so as to maximize their power production. In this problem, the learner has to repeatedly decide about the orientation of the panels so as to find alignments with strong sunshine. Besides the amount of the energy being actually produced in the current alignment, the learner can also possibly base its decisions on measurements of sensors installed on the solar panel. However, the observations generated by these sensors can be of variable quality depending on visibility conditions, the quality of the sensors and the alignment of the panels. Overall, this problem can be seen as a bandit problem with noisy side-observations fitting into our framework, where actions correspond to alignments and the noisy side observations give information about similar alignments.

Formally, the learning protocol (Figure 2.8) additionally assumes the knowledge of the weight of each arc $i \rightarrow j$ in \mathcal{G}_t , which is denoted as $s_{t,(i,j)}$ and assumed to lie in $[0, 1]$. The feedback that the learner in the noisy setting is

$$c_{t,i} = s_{t,(I_t,i)} \cdot \ell_{t,i} + (1 - s_{t,(I_t,i)}) \cdot \xi_{t,i}$$

for every arm i , where $\xi_{t,i}$ is the *observation noise* (c.f. another illustration on Figure 2.7). We assume that each $\xi_{t,i}$ is zero-mean, satisfies $|\xi_{t,i}| \leq R$ for some known constant $R \geq 0$, and is generated independently of all other noise terms and the history of the process.²

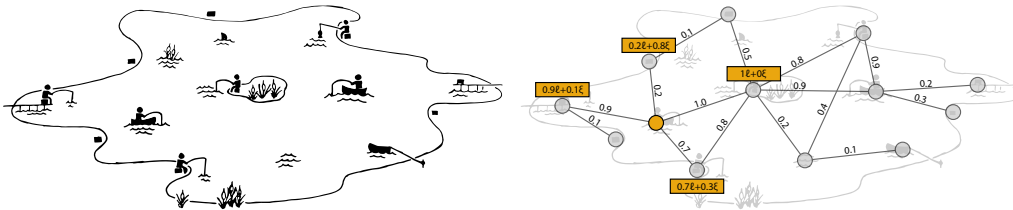


Figure 2.7: *Noisy feedback on a fishing example* (Kocák et al. 2016b; Wu et al. 2015): A fisherman picks a fishing spot daily and gets the yield while imperfectly observing the yields of neighbors.

Intuitively, in the case when the noise level of side observations does not change with time, a possible strategy one can think of is to use only the observations from the *most reliable* sources

²We are mainly interested in the setting where $R = \Theta(1)$, that is, we are neither in the easy case where R is close to zero or the hard one where it may be as large as $\Omega(\sqrt{T})$.

and ignore the rest. Having made the distinction between *reliable* and *unreliable*, the learner could model the observation structure in the framework of Mannor and Shamir (2011), by treating every reliable observation as *perfect*. This approach raises two concerns. First, determining the cutoff for unreliable observations that allows the *most efficient* use of information is a highly nontrivial design choice. As we show later, knowing the *perfect cutoff* would help us to improve performance over the pure bandit setting without side observations. Second, one has to address the *bias* arising from handling every reliable observation as perfect. While one can think of many obvious ways to handle this bias by appropriate weighting observations, none of these solutions are directly compatible with the model of Mannor and Shamir (2011). A central concept in our performance guarantees is a new graph property that we call *effective independence number*, defined as follows:

Definition 2.3.1 Let \mathcal{G} be a weighted directed graph with N nodes and edge weights $s_{i,j}$ bounded in $[0, 1]$. For all $\varepsilon \in [0, 1]$, let $\mathcal{G}(\varepsilon)$ be the (unweighted) directed graph where arc $i \rightarrow j$ is present if and only if $s_{i,j} \geq \varepsilon$ in \mathcal{G} . Letting $\alpha(\varepsilon)$ be the independence number of $\mathcal{G}(\varepsilon)$, the **effective independence number** of \mathcal{G} is defined as

$$\alpha^* = \min_{\varepsilon \in [0,1]} \frac{\alpha(\varepsilon)}{\varepsilon^2}.$$

We first consider an algorithm that bases its decisions on the following estimates of each $\ell_{t,i}$:

$$\widehat{\ell}_{t,i}^{(B)} = \frac{c_{t,i}}{\sum_{j=1}^N p_{t,j} s_{t,(j,i)} + \gamma_t} \quad (2.5)$$

where B stands for *basic*. Here, $\gamma_t \geq 0$ is a so-called *implicit exploration* (or, in short, IX) parameter first used by Kocák et al. (2014a) for decreasing the variance of importance-weighted estimates. Notice that setting $\gamma_t = 0$, makes the estimates above unbiased since

$$\mathbb{E}[c_{t,i} | \mathcal{F}_{t-1}] = \left(\sum_{j=1}^N p_{t,j} s_{t,(j,i)} \right) \cdot \ell_{t,i},$$

where we used our assumption that $\mathbb{E}[\xi_{t,i}] = 0$. Using these estimates in our algorithmic template Exp3 (see Algorithm 1), one would expect to get reasonable performance guarantees. Unfortunately however, we were not able to prove a performance guarantee for the resulting algorithm.

A close examination reveals that the reason for the poor performance of the above algorithm is the large variance of the estimates (2.5) which is caused by including observations from unreliable sources with small weights. One intuitive idea is to explicitly draw the line between reliable and unreliable sources by cutting connections with weights under a certain threshold. This effect is realized by the estimates

$$\widehat{\ell}_{t,i}^{(T)} = \frac{c_{t,i} \mathbb{1}_{\{s_{t,(t,i)} \geq \varepsilon_t\}}}{\sum_{j=1}^N p_{t,j} s_{t,(j,i)} \mathbb{1}_{\{s_{t,(j,i)} \geq \varepsilon_t\}} + \gamma_t}, \quad (2.6)$$

where $\varepsilon_t \in [0, 1]$ is a threshold value and T stands for *thresholded*. We call the algorithm resulting from using the above estimates in Algorithm 1 Exp3-IXt, standing for “Exp3 with Implicit eXploration and Truncated side-observation weights”. Thanks to the thresholding operation, the variance of the loss estimates can be nicely controlled and it becomes possible to prove a strong performance guarantee for Exp3-IXt. Note that if we choose $\varepsilon_t = \arg \min_{\varepsilon \in [0,1]} \frac{\alpha_t(\varepsilon)}{\varepsilon^2}$ for all t , this essentially becomes $\widetilde{\mathcal{O}}(\sqrt{\alpha^*_{\text{avg}} T})$ where $\alpha^*_{\text{avg}} = \frac{1}{T} \sum_{t=1}^T \alpha^*_t$ is the average effective independence number of the sequence of graphs played by the environment. Note however that tuning ε_t can be a very challenging task in practice, since computing independence numbers in general is known to

be NP-hard. Even worse, computing the *effective* independence number of a weighted graph can require computing up to N^2 independence numbers. We propose an adaptive algorithm (**Exp3-WIX**) that does not need to tune this parameter and still manages to guarantee the same regret bound without having to estimate any effective independence numbers. The key element of this algorithm is using loss estimates of the form

$$\widehat{\ell}_{t,i} = \frac{S_{t,(I_t,i)} \cdot C_{t,i}}{\sum_{j=1}^N p_{t,j} S_{t,(j,i)}^2 + \gamma_t}, \quad (2.7)$$

for which we prove the following guarantee.

Theorem 2.3.1 — Regret of Exp3-WIX by Kocák et al. (2016b). For all t , let α_t^* be the effective independence number of \mathcal{G}_t . Then, setting $\eta_t = \sqrt{\frac{\log N}{2(1+R+R^2)(N+\sum_{s=1}^{t-1} Q_s)}}$ and $\gamma_t = R\eta_t$, the regret of **Exp3-WIX** is bounded as

$$R_T = \tilde{\mathcal{O}} \left((1+R) \sqrt{N + \sum_{t=1}^T \alpha_t^*} \right).$$

2.4 Stochastic losses

In this section, we discuss few results for a simpler setting, when the node losses are coming from some fixed distribution. Caron et al. (2012) proposed UCB-N and UCB-MaxN that closely follow UCB, but in addition, they use side observations for better reward estimates (UCB-N) or choose one of the neighboring nodes with a better empirical estimate (UCB-MaxN). These modifications enable to improve the guarantees of UCB, i.e., the regret does not scale with the number of nodes but with the *clique partition number*. Later, Baccapatnam et al. (2014) improved the results of Caron et al. (2012) with LP-based solutions and guarantees scaling with the *minimum dominating set* and Kolla et al. (2016) considered a collaborative setting.

2.4.1 Gaussian losses and side observations

Wu et al. (2015) considered an essentially identical model from Section 2.3 in the stochastic case. In particular, they study partial-observability model for online learning: there, side observations are modeled as zero-mean Gaussian random variables with *variance* depending on the chosen action. It is easy to see that their model and ours can capture exactly the same type of problems as in the adversarial setting: a side observation with zero variance in their model corresponds to a perfect observation with weight 1 while useless noise is equivalently represented by infinite-variance or zero-weight observations. Wu et al. (2015) assume that the losses are i.i.d. Gaussian random variables while the results of Section 2.3 hold without any assumptions made on the sequence of losses. The main contributions of Wu et al. (2015) are (i) a general problem-dependent lower bound on the regret and (ii) algorithms that work under the assumption that all the useful (i.e., finite-variance) side-observations have the same variance. This latter assumption does not use the full strength of the framework where the variance of side observations can vary for different actions.

2.5 Lower bounds and high-probability bounds

While the independence number α can be much lower than the number of nodes N , we may wonder whether it is the *right quantity* describing the *difficulty* of the setting. To support this, Mannor and Shamir (2011) gave an $\Omega(\sqrt{\alpha N})$ bound in the undirected setting for an *unchanging sequence* of graphs, $\mathcal{G}_t = \mathcal{G}$, $\alpha_t = \alpha$. Later, Alon et al. 2013 extended this lower bound to the directed case, still for unchanging sequence of graphs.

While the upper bounds in this chapter were given on the *expected* regret, some algorithms also come with regret guarantees in *high probability*. Alon et al. (2014) gave a high-probability bound for ELP.P, a modified version of ELP that with probability $1 - \delta$ achieves the regret of $\mathcal{O}\left(\sqrt{\log(N/\delta) \sum_{t=1}^T \text{mas}(\mathcal{G}_t)}\right)$, where $\text{mas}(\mathcal{G}_t)$ is the size of the *maximal acyclic subgraph*. While for undirected³ graphs $\text{mas}(\mathcal{G}_t) = \alpha(\mathcal{G}_t)$, for directed graphs $\alpha(\mathcal{G}_t) \leq \text{mas}(\mathcal{G}_t)$ in general and therefore the bound is not as tight. For Exp3-IX, Neu (2015) proved that with probability $1 - \delta$, the cumulative regret of Exp3-IX is bounded by $\tilde{\mathcal{O}}(\sqrt{\alpha N})$, which matches the lower bound of Mannor and Shamir (2011) up to logarithmic factors.

Concerning the *noisy* side observations, Wu et al. (2015) showed an $\Omega(\sqrt{\alpha N}/\varepsilon)$ lower bound on the regret for the special case of graphs, with all weights s_{ij} equal to either 0 or ε . Note that this lower bound matches the upper bound of Exp3-WIX (Theorem 2.3.1), since in that case $\alpha^* = \alpha/\varepsilon^2$.

2.6 Perspectives for side observations

In this section, we describe new challenges, related settings, and open problems for graph bandit learning with side observations.

2.6.1 Beyond bandits

Besides the side observation models mentioned above, several other partial-observability models have been considered in the literature. The most general of these settings is the *partial-monitoring* framework considered by Bartók et al. (2014) and Bartók et al. (2011). Unlike the side observation model, this framework is most useful for identifying and handling feedback structures that are *more restrictive* than bandit feedback. In contrast, learning with side observations deals with feedback structures that are strictly more expressive than plain bandit feedback. Similarly to Bartók et al. (2011), the recent work of Alon et al. (2015) also considers a generalization of the partial-observability models of Mannor and Shamir (2011) and Alon et al. (2013) that may be more restrictive than bandit feedback.

Specifically, Alon et al. (2015) consider directed graphs with possible *self-loops*. For a particular node, a self-loop means that whenever this node is selected, its loss is observed. Therefore, online learning on graphs with side observations, as defined by Mannor and Shamir (2011) and considered above is a special case when all the self-loops are always present. More restrictive feedback schemes emerge when some of the self-loops are not available, which means that the learner does not observe the loss of the chosen action, but still occurs this loss. Similarly to Bartók et al. (2011), they found that there are 3 classes of problems with $\tilde{\Theta}(\sqrt{T})$, $\tilde{\Theta}(T^{2/3})$, and $\tilde{\Theta}(T)$ regret and interestingly provide a *complete characterization* of the settings classifying all possible graphs in these three categories. Furthermore, a generalized version of Exp3.G (Section 2.2.2) can be used to attain these rates.

Cesa-Bianchi et al. (2016) study yet another learning setting when the nodes cooperate to solve a nonstochastic bandit problem by communicating up to d hops on the graph. Their Exp3-Coop algorithm is shown to scale with $\alpha_{\leq d}$, which is the independence number of the d -th power of the connected communication graph \mathcal{G} . Furthermore, Ghosh and Prügel-Bennett (2015) study a quite non-standard setting with Ising graph model.

Before to research in graph bandits and the quest for tight finite-time regret bounds, there was a prior work in economics and social sciences that studied the asymptotic convergence of learning for specific social models (Bala and S. Goyal 1998; Bala and S. Goyal 2001; Ellison and Fudenberg 1993; Gale and Kariv 2003).

³where we consider two edges between the same nodes going the opposite direction

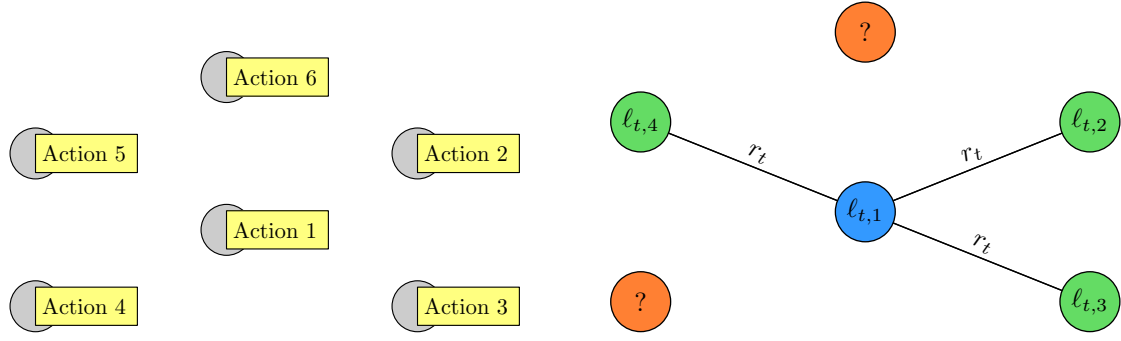


Figure 2.9: **Left:** The learner selects one of the actions (e.g., Action 1). **Right:** The nature generates an Erdős-Rényi graph with parameter r_t , where r_t can be chosen by an adversary.

2.6.2 Graph generators

One of the main practical drawback on the settings and algorithms presented in this chapter is the need to see some parts of the graph, at least after the action was chosen. Indeed, all previous algorithms for the studied setting (Alon et al. 2013; Kocák et al. 2014a; Mannor and Shamir 2011) require the environment to reveal a substantial part of a graph, at least after the side observations have been revealed. Specifically, these algorithms require the knowledge of the *second neighborhood* (the set of neighbors of the neighbors) of the chosen action in order to update their internal loss estimates. On the other hand, they are able to handle arbitrary graph structures, potentially chosen by an adversary and prove performance guarantees expressed using graph properties based on cliques or independence sets. In fact, it is difficult to get rid of this constraint, since Cohen et al. (2016) show that achieving nontrivial advantages from side observations may be impossible without perfectly known side-observation graphs when an adversary is allowed to pick *both* the losses and the side-observation graphs. However, the situation is easier if we know something more about how \mathcal{G}_t is generated.

Erdős-Rényi side-observation graphs

Erdős-Rényi (ER) graphs (Erdős and Rényi 1959) are well studied random graphs where each edge is generated uniformly at random with probability r (Figure 2.9). If this probability is fixed, but a new graph \mathcal{G}_t can be generated every round, then the regret of Exp3-SET is of $\mathcal{O}\left(\frac{2(\log N)T(1-(1-r)^N)}{r}\right)$ (Alon et al. 2013). Furthermore, generalizing the lower bound of Mannor and Shamir (2011), Alon et al. (2013) also proved a $\Omega\left(\sqrt{T/r}\right)$ lower bound for this setting in the case of a fixed graph. However, Exp3-SET still needs to have the knowledge of r and to have the parts of the graph revealed after the actions. An interesting direction would be an algorithm that would not require this knowledge, since the probability of the side observation is r . Therefore, we can strive for an algorithm with $\tilde{\mathcal{O}}\left(\sqrt{T/r}\right)$ regret in the fixed r case and $\tilde{\mathcal{O}}\left(\sqrt{\sum_{t=1}^T(1/r_t)}\right)$ in case of changing r . Note that when $r < 1/N$, these bound are worse than ignoring all side observations (the case of Exp3) and therefore the most interesting would be a procedure that does not do worse than Exp3.

In the case if r_t is not to small, we provided **Exp3-Res** (Kocák et al. 2016a), an algorithm that can efficiently estimate the losses without explicitly estimating r_t . The main challenge in our setting is leveraging side observations *without knowing* r_t . Had we had access to the exact value of r_t , we would be able to define the following estimate of $\ell_{t,i}$:

$$\hat{\ell}_{t,i}^* = \frac{O_{t,i}\ell_{t,i}}{p_{t,i} + (1 - p_{t,i})r_t}$$

It is easy to see that the loss estimates defined this way are unbiased in the sense that $\mathbb{E}[\widehat{\ell}_{t,i} | \mathcal{F}_{t-1}] = \ell_{t,i}$ for all t and i . It is also straightforward to show that an appropriately tuned instance of the Exp3 algorithm of Auer et al. (2002b) fed with these loss estimates is guaranteed to achieve a regret of $\mathcal{O}(\sqrt{\sum_t (1/r_t) \log N})$ (see also Seldin et al. 2014). or any fixed t, i , we now describe an efficiently computable surrogate $G_{t,i}$ for the geometrically distributed random variable $G_{t,i}^*$ with parameter $o_{t,i}$ that will be used for constructing our loss estimates. In particular, our strategy will be to construct several independent copies $\{O'_{t,i}(k)\}$ of $O_{t,i}$ and choosing $G_{t,i}$ as the index k of the first copy with $O'_{t,i}(k) = 1$. It is easy to see that with infinitely many copies, we could exactly recover $G_{t,i}^*$; our actual surrogate is going to be weaker thanks to the smaller sample size. For clarity of notation, we will omit most explicit references to t and i , with the understanding that all calculations need to be independently executed for all pairs t, i .

Let us now describe our mechanism for constructing the copies $\{O'(k)\}$. Since we need independence of $G_{t,i}$ and $O_{t,i}$ for our estimates, we use only side observations from actions $[N] \setminus \{I_t, i\}$. First, let's define σ as a uniform random permutation of $[N] \setminus \{I_t, i\}$. For all $k \in [N-2]$, we define $R(k) = O_{t,\sigma(k)}$. Note that due to the construction, $\{R(k)\}_{k=1}^{N-2}$ are pairwise independent Bernoulli random variables with parameter r_t , independent of $O_{t,i}$. Furthermore, knowing $p_{t,i}$ we can define $P(1), \dots, P(N-2)$ as pairwise independent Bernoulli random variables with parameter $p_{t,i}$. Using $P(k)$ and $R(k)$ we define the random variable $O'(k)$ as

$$O'(k) = P(k) + (1 - P(k))R(k)$$

for all $k \in [N-2]$. Using independence of all previously defined random variables, it is easy to check that the variables $\{O'(k)\}_{k=1}^{N-2}$ are pairwise independent Bernoulli random variables with expectation $o_{t,i} = p_{t,i} + (1 - p_{t,i})r_t$. Now we are ready to define $G_{t,i}$ as

$$G_{t,i} = \min \{k \in [N-2] : O(k)' = 1\} \cup \{N-1\}.$$

We can show that $G_{t,i}$ follows a truncated geometric law in the sense that

$$\mathbb{P}[G_{t,i} = m] = \mathbb{P}[\min \{G_{t,i}^*, N-1\} = m]$$

holds for all $m \in [N-1]$. Using all this notation, we construct an estimate of $\ell_{t,i}$ as

$$\widehat{\ell}_{t,i} = G_{t,i} O_{t,i} \ell_{t,i}. \tag{2.8}$$

The rationale underlying this definition of $G_{t,i}$ is rather delicate. First, note that $p_{t,i}$ is deterministic given the history \mathcal{F}_{t-1} and therefore, does not depend on $O_{t,i}$. Second, $O_{t,i}$ is also independent of $O_{t,j}$ for $j \notin \{i, I_t\}$. As a result, $G_{t,i}$ is independent of $O_{t,i}$, and we can use the identity $\mathbb{E}_t[G_{t,i} O_{t,i}] = \mathbb{E}_t[G_{t,i}] \mathbb{E}_t[O_{t,i}]$. Using the estimates from Equation 2.8 in Line 12 of Algorithm 1, we get the **Exp3-Res** algorithm. The next theorem states our main result concerning **Exp3-Res** with an adaptive learning rate.

Theorem 2.6.1 — Regret of Exp3-Res by Kocák et al. (2016a). Assume that $r_t \geq \frac{\log T}{2N-2}$ holds

for all t and set $\eta_t = \sqrt{\frac{\log N}{N^2 + \sum_{s=1}^{t-1} \sum_{i=1}^N p_{s,i}(\widehat{\ell}_{s,i})^2}}$. Then, the expected regret of **Exp3-Res** satisfies

$$R_T \leq 2 \sqrt{\left(N^2 + \sum_{t=1}^T \frac{1}{r_t} \right) \log N} + \sqrt{T}.$$

The most obvious question and currently an open problem is whether it is possible to remove our assumptions on the values of r_t . We can only give a definite answer in the simple case when all r_t -s

are identical: In this case, one can think of simply computing the empirical frequency \hat{r}_t of all previous side observations in round t to estimate the constant r .

Besides Erdős–Rényi graphs, another direction would be the extension of the known results to side information in Barabási-Albert (1999) or Watts-Strogatz (1998) model, or other models better suited for some real-world graphs (e.g., social networks).

Side observations in the communities

One typical target scenario for the setting in this chapter is advertising on social networks, where the advertiser chooses a target user and besides their feedback receives (as side observations) also the feedback of their contacts. Social networks are often modeled as a set of (overlapping) communities (Figure 2.10) and therefore an extension is to consider an assumption that graphs we deal with have a community structure. First, we can consider the communities as yet another model for random graphs. The most studied model is the *stochastic block model* and its variants (Girvan and Newman 2002). Second, we may have access to the community model and consider the case where each community gives side observations with their own probabilities, which are unknown to the learner.

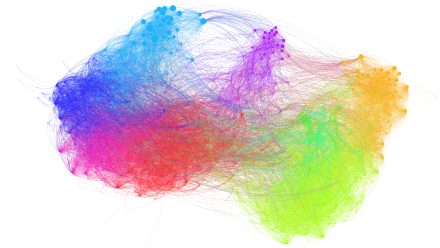



Figure 2.10: Communities.



3. Influence maximization

Product placement is another marketing application that we target with graph bandits. An advertiser can offer a product to some users in a hope that they will recommend the product to their contacts, i.e., to the neighboring nodes in a social network. The advertiser then observes the set of contacts that these users have influenced and that have bought the product. The objective of the advertiser is to target *influential users*, the nodes of the graph whose influence is the most important. Ideally, the advertiser would only offer products to the users with maximum influence.

Furthermore, there are many models of influence and some of the known ones were introduced in the seminal work on spreading the influence through a social network (Kempe et al. 2003; Kempe et al. 2015). In this chapter, we focus on *local influence*, where a node on the graph influences only its *immediate neighborhood* and outline the road for more *global models*.

We finished the previous chapter by stating that most of the existing approaches for active learning on graphs assume that either the *entire graph* is known in advance, or at least that a substantial *part of the graph* is revealed to the learner after it selected the node. Typically, the algorithms require at least the knowledge of the set of neighbors of the neighbors of the nodes (*second neighborhood*). This knowledge of the graph is crucial for existing learning algorithms (Alon et al. 2015; Alon et al. 2013; Buccapatnam et al. 2014; Caron et al. 2012; Cesa-Bianchi et al. 2013; Gentile et al. 2014; Gu and Han 2014; Kocák et al. 2014a; Mannor and Shamir 2011; Valko et al. 2014; Yu and Mannor 2011) to help them learn faster than in the case if no structure existed. However, in some realistic scenarios, the graph information is *not available* to the learner beforehand. Typically, the operator of the social network would not freely reveal the social links and therefore the graph is not known to the advertiser. On the other hand, for instance, in the advertising example presented above, the advertiser has some local access to the social network in the sense that they can get information of the set of users that were influenced to purchase products through the other targeted customers. This information can be gathered through *promotional codes* when the goal is *product purchase* or through *likes* in an *information campaign* (Caron et al. 2012).

However, the existing graph bandit approaches do not allow to treat this scarce side information setting. Therefore, with the known tools, one can either (i) first thoroughly explore the graph and then apply existing graph bandit strategies, or (ii) forget about the underlying graph structure and

apply existing multi-arm bandit algorithms to the nodes of the graph. In both cases, it is necessary that the learner substantially explores the graph and therefore *samples many nodes*, if not all of them. This is not very reasonable, for instance, in our marketing example, since graphs corresponding to social networks are usually large. Moreover, the advertiser is unlikely to have a large enough budget to target all the nodes of the graph in order to learn which ones are the most influential.

3.1 Local influence and revelation bandits

Let \mathcal{G} be a graph with N nodes. When a node i is selected, it can influence the nodes of \mathcal{G} , including itself. Node i influences each node j with *fixed but unknown* probability $p_{i,j}$ (Figure 3.1). Let $\mathbf{M} = (p_{i,j})_{i,j}$ be the $N \times N$ matrix that represents \mathcal{G} . We consider the following online, active setting. At each round (time) t , the learner chooses a node k_t and observes which nodes are influenced by k_t , i.e., the set $S_{k_t,t}$ of influenced nodes is *revealed*. Given a budget of T rounds, the objective is to maximize the number of *influences* that the selected node exerts. Formally, our goal is to find the strategy maximizing the performance

$$\text{Reward}_T = \sum_{t=1}^T |S_{k_t,t}|.$$

The *influence* of node k , i.e., the expected number of nodes that node k exerts influence on, is by definition

$$r_k = \mathbb{E}[|S_{k,t}|] = \sum_{j \leq N} p_{k,j}.$$

We also define the *dual influence* of node k as

$$r_k^\circ = \sum_{j \leq N} p_{j,k}.$$

This quantity is the expected number of nodes that exert influence on node k . For an undirected graph \mathcal{G} , \mathbf{M} is symmetric and $r_k^\circ = r_k$. However, in general, this is not the case, but we assume that the influence is up to a certain degree mutual. In other words, we assume that if a node is very influential, it also is subject to the influence of many other nodes.

As the performance measure, we compare any *adaptive strategy* for this setting with the optimal oracle that knows \mathbf{M} . The oracle strategy always chooses one of the most influential nodes, which are the nodes whose expected number of influences r_k is the largest. We call one of these node k^* , such that

$$k^* = \arg \max_k \mathbb{E} \left[\sum_{t=1}^T |S_{k,t}| \right] = \arg \max_k T r_k.$$

Let the reward of this node be

$$r_* = r_{k^*}.$$

Then, its expected performance, if it consistently sampled k^* over T rounds, is equal to

$$\mathbb{E}[\text{Reward}_T^*] = T r_*.$$

The expected *regret* of any adaptive strategy that is unaware of \mathbf{M} , with respect to the oracle strategy, is defined as the expected difference of the two,

$$\mathbb{E}[R_T] = \mathbb{E}[\text{Reward}_T^*] - \mathbb{E}[\text{Reward}_T].$$

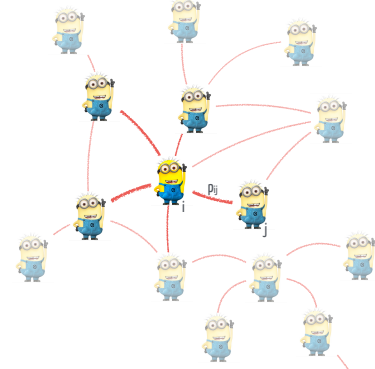


Figure 3.1: Influence probability $p_{i,j}$.

Dually, we define r_\star° as the average number of influences received by the most influenced node,

$$r_\star^\circ = \max_k r_k^\circ.$$

First, note that the minimax-optimal rate in this setting is the same as in the restricted information case, when we ignore the identity of the influenced nodes and only use the number of them as a reward. To see that, one can, for instance, consider a network composed of isolated nodes with only a very small clique of most influential nodes, connected only to each other. Another example is a graph where the fact of being influential is uncorrelated with the fact of being influenced and where, for instance, the most influential node is not influenced by any node. Therefore, when $T \leq N$, there is no adaptive strategy in a minimax sense, also in this unrestricted setting we just defined.

However, the cases where the identity of the influenced nodes does not help, are somewhat pathological. Intuitively, they correspond to cases where the graph structure is not very informative for finding the most influential node. This is the case when there are many isolated nodes, and also in the case where observing nodes that are very influenced does not provide information on these nodes' influence. In many typical and more interesting situations, this is not the case. First, in these problems, the nodes that have high influence are also very likely to be subject being influenced, for instance, many interesting networks are symmetric and then it is immediately the case. Second, in realistic graphs, there is typically a small portion of the nodes that are noticeably more connected than the others (Barabási and Albert 1999).

In order to rigorously define these nondegenerate cases, let us first define the function D that controls the number of nodes with a given *dual gap*, i.e., a given suboptimality with respect to the most influenced node.

$$D(\Delta) = |\{i \leq N : r_\star^\circ - r_i^\circ \leq \Delta\}|.$$

The function $D(\Delta)$ is a nondecreasing quantity dual to the arm gaps. Note that $D(r) = N$ for any $r \geq r_\star^\circ$ and that $D(0)$ is the number of most influenced nodes. We now define the *problem dependent* quantities that express the difficulty of the problem and allow us to state our results.

Definition 3.1.1 We define the **detectable horizon** as the smallest integer $T_\star > 0$ such that

$$T_\star r_\star^\circ \geq \sqrt{D_\star T r_\star^\circ},$$

when such T_\star exists and $T_\star = T$ otherwise. Here, D_\star is the **detectable dimension** defined as

$$D_\star = D(\Delta_\star),$$

where the **detectable gap** Δ_\star is defined as

$$D_\star \stackrel{\text{def}}{=} 16 \sqrt{\frac{r_\star^\circ N \log(TN)}{T_\star} + \frac{144N \log(TN)}{T_\star}}.$$

R From the definitions above, the detectable dimension is the D_\star that corresponds to the smallest integer $T_\star > 0$ such that

$$T_\star r_\star^\circ \geq \sqrt{D \left(16 \sqrt{\frac{r_\star^\circ N \log(TN)}{T_\star} + \frac{144N \log(TN)}{T_\star}} \right) T r_\star^\circ},$$

or $D_\star = N$ if such T_\star does not exist. It is therefore a well defined quantity. Moreover, since D is nondecreasing and $D(0)$ is the number of most influenced nodes, then D_\star converges to the number of most influenced nodes as T tends to infinity.

Finally let us write the influential-influenced gap as

$$\varepsilon_\star \stackrel{\text{def}}{=} r_\star - \max_{k \in \mathcal{D}^\circ} r_k,$$

where $\mathcal{D}^\circ \stackrel{\text{def}}{=} \{i : r_i^\circ = \max_k r_k^\circ\}$. The quantity ε_\star quantifies the gap between the most influential node overall vs. the most influential node in the set of most influenced nodes.

R The quantity ε_\star is small when one of the most influenced node is also very influential. It is exactly zero when one of the most influential nodes happens to also be one of the most influenced nodes. For instance, the case $\varepsilon_\star = 0$ appears in undirected social network models with mutual influence.

The graph structure is helpful when the D function decreases quickly. To get an intuition, consider a star-shaped graph which is the most helpful and can have $D_\star = 1$ even for a small T . On the other hand, a bad case is a graph with many small cliques. The worst case is where all nodes are disconnected except 2, where D_\star will be of order N even for a large T .

The detectable dimension D_\star is a problem dependent quantity that represents the *complexity of the problem* instead of N . In real networks, D_\star is typically smaller than the number of nodes N . As our analysis will show, D_\star represents the number of nodes that we can *efficiently extract* from the mass of the N nodes in less than T rounds of the time budget. Our *bandit revelator* algorithm, **BARE** (Carpentier and Valko 2016), starts by the *global exploration* phase and extracts a subset of cardinality less than or equal to D_\star , that contains a very influential node, that is at most ε_\star away from the most influential node. **BARE** does this extraction *without scanning all the N nodes*, which could be impossible, anyway since we do not restrict to $N \leq T$. In the subsequent *bandit* phase, **BARE** proceeds with scanning this smaller set of selected nodes to find the most influential one.

We now state our main theoretical result that proves a bound on the regret of **BARE**.

Theorem 3.1.1 — Regret of **BARE by Carpentier and Valko (2016).** In the unrestricted local influence setting with information on the neighbors, **BARE** satisfies, for a constant $C > 0$,

$$\mathbb{E}[R_T] \leq C \min(r_\star T, D_\star r_\star + \sqrt{r_\star T D_\star} + T \varepsilon_\star).$$

While detectable dimension D_\star behaves as we expect, it does not seem to be directly linked with some previously known graph concept (as it was the case for the side observations and independence number). In fact, the graph-dependent only quantity is the function $D(\Delta)$, that quantifies the amount of Δ -suboptimal most influenced nodes. However, the detectable dimension itself is tied to the bandit problem by essence (and the constants are due to the Bernstein bound) — it is the quantity that realizes the optimal tradeoff between the regret suffered during the global exploration phase, and the regret suffered during the bandit phase. To support this claim, we give a lower bound that features this quantity. Notice that the influential-influence gap also appears in it.

Theorem 3.1.2 — Lower bound for local influence setting by Carpentier and Valko (2016).

Let $N \geq CT > 0$ where $C > 0$ is a universal constant. Consider the set of local influence setting and the set of all problems that have maximal influence bounded by r , detectable dimension smaller than $D \leq N/2$ and influential-influence gap smaller than ε . Then the expected regret of the best possible algorithm in the worst case of these problems is lower bounded as

$$C'' \min(rT, Dr_\star + \sqrt{rTD} + T\varepsilon),$$

where C'' is a universal constant.

Large scale setting

The quantity D_* and BARE become particularly appealing when we consider an interesting practical situation with a large number of graph nodes. For instance, even in a medium-sized social network, the advertiser would not have enough budget to target all the users and discover the most influential one, i.e., $T \leq N$. Notice again, that in the restricted setting, the regret of bandit strategies in this problem for $T \ll N$ is of order Tr_* , which is larger than the regret of BARE.

3.2 Perspectives of bandit influence maximization

In this section, we outline some extensions of the simple model of influence described above.

3.2.1 Global models of influence

In Section 3.1, we discussed the local influence model. In computational social sciences, we usually consider more involved, global models of the influence spread over a social graph. The most known and studied are the models described in the seminal paper of Kempe et al. (2003), in particular, the *independent cascade* model. In this model, we consider a set of seed nodes $A_0 \subseteq \mathcal{V}$ and a probability p_{ij} associated with each edge. Independent cascade model defines an activation process of nodes, where at the beginning, all nodes in A_0 are active and subsequently every node i can activate its neighbor j with probability p_{ij} once, independent of the history of the process. This process runs until no more activations are possible. Given the set of probabilities, $\{p_{ij}\}_{ij}$, the goal in the (offline) influence maximization problem is to find such A_0 that maximizes the expected number of influenced nodes. Obviously, this property is trivially maximized for the whole node set $A_0 = \mathcal{V}$, but we are typically interested in the $|A_0| \leq k$, where k is coming from the budget constraint of how many people we can afford to reach. This *offline* problem is NP-hard, but as the (expected) number of influence nodes is a submodular set function, it can be approximated within the factor of $1 - 1/e$ (Kempe et al. 2003).

Similarly to the local influence (Section 3.1), in the bandit setting, the set of activation probabilities $\{p_{ij}\}_{ij}$ are *unknown* to the learner. In this simplest case, $k = 1$ and we are interested in selecting a single, most influential node. In general, $k \in [N - 1]$ and this is an instance of combinatorial bandits. We can consider several feedback settings:

1. *full bandit*: the learner only observes the *number* of influenced nodes
2. *node semi-bandit*: learner observes the *identity* of the influenced nodes
3. *edge semi-bandit*: learner observes the *identity* of the activated edges

Notice that in the edge semi-bandit setting, we observe a sample (one) from $\text{Bernoulli}(p_{ij})$ for each activated edge. Moreover, we also receive a sample (zero) from $\text{Bernoulli}(p_{ij})$ for each nonactivated edge, when at least one of i or j nodes were activated. The node semi-bandit feedback is more challenging, since we do not observe the activation edges, and therefore we do not know what was the activation path for that node, which makes the estimation of p_{ij} nontrivial.

For the full bandit feedback and $k = 1$, we can obtain results similar to those as in the restricted setting considered in the local influence model. On the other hand, the understanding of the semi-bandit feedback for the influence maximization problem is still an open problem. Nonetheless, we comment on some recent attempts and results. Recently, Lei et al. (2015) investigated the combinations of offline influence maximization approaches with multi-arm bandit strategies for the online influence maximization in the edge semi-bandit case. Lei et al. (2015) tried several combinations of bandit techniques (ϵ -greedy, confidence-based methods) and empirically showed that their methods perform well, however, they did not provide any guarantees or analysis. Chen et al. (2016) also considered combinatorial edge semi-bandit case and showed that the reward function of this problem is a special case of their general combinatorial semi-bandit case satisfying *monotonicity* and *bounded-smoothness* conditions. Therefore, their algorithm (CUCB) and analysis

apply. However, their analysis is general and distribution-dependent only and it is not clear how it relates to the structure of the graph. Furthermore, both of their gap-dependent and gap-free bounds are problematic because they depend on the reciprocal of the minimum observation probability p^* of an edge. Consider a line graph with L edges where all edge weights are 0.5. Then $1/p^*$ is 2^{L-1} . To avoid this problem, we proposed (Wen et al. 2016) **IMLinUCB**, a linear UCB-like algorithm for edge semi-bandits that permits linear generalization and is suitable for large-scale problems. We bounded the regret of **IMLinUCB** when the structure of the network is a *forest* (Wen et al. 2016). Our regret bounds are polynomial in all quantities of interest; reflect the structure and activation probabilities of the network; and do not depend on inherently large quantities, such as the reciprocal of the minimum probability of being influenced and the cardinality of the action set. The forest is important in practice because influence maximization in general graphs is computationally expensive, and known scalable approximations use forests to evaluate only most influential paths, such as in the maximum influence arborescence (MIA) model (Chen et al. 2010). Furthermore, Vaswani et al. (2015) consider the more difficult, *node* semi-bandit setting. In this setting, however, it is unknown which edge was alive and should have its estimate updated. Vaswani et al. (2015) decide to update one of the edges that could have been alive uniformly at random. It is not clear whether it is possible to do better and also, what is the equivalent of *detectable dimension* for this model. Another direction is to estimate the influence function using the recent results studying learnability of influence in networks (Narasimhan et al. 2015). Finally, the problem gets even more challenging when we allow the influence probabilities to change (Bao et al. 2016), when we allow the seed set to be chosen adaptively (Vaswani and Lakshmanan 2016), or when we consider a continuous model (Farajtabar et al. 2016). To sum up, bandit influence maximization under global models remains a very interesting open problem.

3.2.2 Crawling bandits

In Chapters 1 and 2, all methods needed to have access to parts of the graph for various learning reasons. In the present chapter, we lifted the assumption on the knowledge of the edge set and the learner had to also *estimate the graph structure* in order to act on it. Yet the learner was allowed to choose *any node at any round*. In a more challenging case, even this possibility can be *restricted*.

As we mentioned before, the inability of the learner to access the full graph as desired can come from some external factors. In the context of advertising in social networks, the social network provider can have reasons to conceal the social graph: privacy, business advantage, or intention of charging for this information. This poses an additional challenge for the learner who can only see (some) neighbors of the previously chosen nodes. Such process resembles *crawling* the websites through the links to collect some information or discover interesting new sites. Singla et al. (2015) formalizes a specific set of these constraints for general *utility functions* using the parameters l_{deg} and l_{val} , where l_{deg} quantifies *selectability* of the new nodes and l_{val} the *observation possibility* of the new nodes and notes that in real-world social networks (such as Facebook or LinkedIn), the visibility is usually restricted to $l_{\text{deg}} = 1$ and $l_{\text{val}} = 1$ due to privacy settings. This means that the learner can typically only see and access (select in the next step) the local (1-hop) neighborhood of the nodes already selected. Singla et al. (2015) uses this restriction parametrization for specific set discovery problems. It is an open problem what algorithm would be optimal for (cumulative) regret minimization. This setting is also related to the *volatile multi-armed bandits* where the set of possible arms changes (Bnaya et al. 2013).

Note that the constraints on the visibility of the graphs are not only applicable in the influence maximization setting, they are relevant in other graph bandits, for instance in learning with side observations (Chapter 2).



Stochastic bandits in large structured domains

4	Kernel bandits	49
4.1	Kernelized UCB	
4.2	Analysis of <code>KernelUCB</code>	
4.3	Relationship with <code>GP-UCB</code>	
4.4	Perspectives of bandits for stochastic processes	
5	Polymatroid bandits	55
5.1	Optimization on polymatroids	
5.2	Combinatorial optimization on polymatroids	
5.3	Learning model	
5.4	The <code>OPM</code> algorithm	
5.5	Discussion and perspectives of polymatroid bandits	
6	Bandits for function optimization	63
6.1	Near-optimality dimension independent of a semi-metric	
6.2	Hierarchical optimistic optimization: <code>HOO</code>	
6.3	Unknown function smoothness: <code>StoS00</code>	
6.4	Parallel optimistic optimization: <code>P00</code>	
6.5	Applicability and perspectives of bandit function optimization	
7	Infinitely many armed bandits	71
7.1	Learning setting	
7.2	Lower bounds	
7.3	<code>SiRI</code> and its upper bounds	
7.4	Extensions of <code>SiRI</code>	
	References	77

The whole previous part was dedicated to settings where the actions (arms) are the graph nodes. Not all action spaces naturally form a graph and in this part we focus on other *structured spaces*. In Chapter 4, we describe the frequentist analysis of kernelized bandits (Valko et al. 2013b), closely related to Gaussian process bandits (Srinivas et al. 2010). Kernelized bandits are a simple extension of linear bandits to reproducing kernel Hilbert spaces (RKHS). In Chapter 5, we consider *polymatroid bandits* (Kveton et al. 2016), that generalize the notion of linear independence to other structures, where the optimization over combinatorial action spaces can be done efficiently (in the offline case) using the simple Greedy algorithm.

While kernelized and polymatroids bandits are instances of discrete action spaces, in the rest of this part we give examples of a structure in *continuous* ones. First, in Chapter 6, we apply bandit strategies to *black-box function optimization* with noisy evaluations, where the action space is a (bounded) continuous domain of some unknown function f . The structure of rewards in this setting is the *smoothness* around one of the optima of f . However, in the most general setting, we treat the case when this smoothness is *unknown* to the learner and we show that we are able to provide almost the same guarantees on the error (*simple regret*) as if this smoothness was available. Second, in Chapter 7 we look into another bandit setting with continuous arm set, but this time with *no topological or metric assumptions* between the arms. In other words, no arm can give any information about any other arm. This setting was formalized by Berry et al. (1997) as *infinitely many arms* bandits and we focus on the *simple regret* in this setting, same as in Chapter 6.

In the previous graph bandit part, the common thread was the study of graph-dependent quantities (independence number, detectable dimension, number of relevant eigenvectors, ...) for different settings that embodied different difficulties of the problems. We studied algorithms that took advantage of the graph and were able to get faster rates as functions of these graph-dependent quantities instead of the number of nodes N . Our intention in this part is very similar. What are the *sizes of action sets* considered here? First, the kernelization of linear bandits in kernel bandits takes the dependence on the dimension D of the context to the dimension of RKHS, that is possibly infinite. Second, the space of actions in polymatroid bandits is combinatorial (possibly exponential) in the number of items. Finally, in both bandits for function optimization and infinitely many arm bandits, the arms form a continuous set. Henceforth, while in graph bandit part, we had a choice of ignoring the graph structure, treat the settings as multi-arms bandits and get a (likely worse) dependence on the number of nodes N ; taking the same path for the settings considered in this part and ignoring the present structure would be *hopeless*. As a consequence, our quest is to find the appropriate problem-dependent quantities also for the large structure settings of this part.

In the case of kernelized bandits, we define a notion of effective dimension, measuring the *decay of eigenvalues* of the covariance matrix in kernel regression. Next, for polymatroid bandits, we show an algorithm whose regret scales with the *rank of the polymatroid* (matching the lower bound in the matroid case). Furthermore, in the black-box function optimization setting, we consider the *near-optimality dimension*, which measures the complexity of the optimization problem. Finally, in the case of infinitely many arm bandits, we give an algorithm optimizing simple regret with the near-optimal guarantees, that depend on a parameter β , characterizing the *distribution of the near-optimal arms*, same β as in the cumulative regret version of Berry et al. (1997).

4. Kernel bandits

This chapter considers a generalized version of the setting of spectral bandits (Section 1.1). Unlike in linear bandits (Auer 2002) we avoid a possibly costly feature-engineering step by assuming that we have access to the similarities between actions' contexts and that the expected reward is an *arbitrary* linear function of the contexts' images in the related reproducing kernel Hilbert space (RKHS). In the following, we show how to derive `KernelUCB` by directly kernelizing the `LinUCB` algorithm. In contrast, GP-UCB is motivated from experimental design. Our derivation is the combination of the kernel trick (Shawe-Taylor and Cristianini 2004) and the kernelized version of the Mahalanobis distance (Haasdonk and Pekalska 2010).

4.1 Kernelized UCB

Kernel methods assume that there exists a mapping $\phi : \mathbb{R}^D \rightarrow \mathcal{H}$ that maps the data to a (possibly infinite dimensional) Hilbert space in which a linear relationship can be observed. We call \mathbb{R}^D the *primal space* and \mathcal{H} the associated *reproducing kernel Hilbert space* (RKHS). We use matrix notation to denote the inner product of two elements $h, h' \in \mathcal{H}$, i.e., $h^\top h' \stackrel{\text{def}}{=} \langle h, h' \rangle_{\mathcal{H}}$ and $\|h\| = \sqrt{\langle h, h \rangle_{\mathcal{H}}}$ to denote the RKHS norm. From the mapping ϕ we have the *kernel function*, defined by:

$$k(x, x') \stackrel{\text{def}}{=} \phi(x)^\top \phi(x'), \quad \forall x, x' \in \mathbb{R}^D,$$

and the *kernel matrix* of a data set $\{x_1, \dots, x_t\} \subset \mathbb{R}^D$ given by $\mathbf{K}_t \stackrel{\text{def}}{=} \{k(x_i, x_j)\}_{i, j \leq t}$. For our nonlinear contextual bandit model we assume the existence of a ϕ for which there exists a $\theta^* \in \mathcal{H}$ such that:

$$\mathbb{E}(r_{a,t} | x_{a,t}) = \phi(x_{a,t})^\top \theta^*.$$

We also let $y_t \stackrel{\text{def}}{=} \{r_{a_1,1}, \dots, r_{a_t,t}\}^\top$ and $X_t \stackrel{\text{def}}{=} \{x_{a_1,1}, \dots, x_{a_t,t}\}^\top$. Taking $a_t^* \stackrel{\text{def}}{=} \arg \max_{a \in \mathcal{A}} \{\phi(x_{a,t})^\top \theta^*\}$ we can define the regret as usual. Note that when $\phi \equiv \text{Id}$, we recover the linear bandit case.

To obtain the upper confidence bounds we derive prediction and width estimators for the expected rewards. `LinUCB` uses estimators built from ridge regression in the primal. Since we assume that our model is linear in the RKHS we show how to build estimators from ridge regression

in \mathcal{H} . By deriving equivalent dual forms which involve only entries of the kernel matrix we avoid working directly in the possibly infinite dimensional RKHS.

First, we take the prediction estimator to be of the form $\hat{\mu}_{a,t+1} = \phi(x_{a,t+1})^\top \theta_t$ where θ_t is the minimizer of the regularized least squares loss function,

$$\mathcal{L}(\theta) = \gamma \|\theta\|^2 + \sum_{i=1}^{t-1} (r_i - \phi(x_i)^\top \theta)^2. \quad (4.1)$$

We derive a representation of this estimator involving only kernels between context vectors. We denote $\Phi_t = [\phi(x_1)^\top, \dots, \phi(x_{t-1})^\top]^\top$. Note that the solution of the minimization problem $\theta_t \stackrel{\text{def}}{=} \min_{\theta \in \mathcal{H}} \mathcal{L}(\theta)$ satisfies

$$(\Phi_t^\top \Phi_t + \gamma I) \theta_t = \Phi_t^\top y_t.$$

Rearranging this equation we obtain

$$\theta_t = \Phi_t^\top \alpha_t, \quad (4.2)$$

where $\alpha_t = \gamma^{-1}(y_t - \Phi_t \theta_t) = \gamma^{-1}(y_t - \Phi_t \Phi_t^\top \alpha_t)$, which implies that $\alpha_t = (\mathbf{K}_t + \gamma I)^{-1} y_t$. Finally, denoting $k_{x,t} \stackrel{\text{def}}{=} \Phi_t \phi(x) = [k(x, x_1), \dots, k(x, x_{t-1})]^\top$ we get

$$\hat{\mu}_{a,t} = \mathbf{k}_{x_{a,t}}^\top (\mathbf{K}_t + \gamma I)^{-1} y_t. \quad (4.3)$$

While the computation of θ_t using (4.2) would require evaluating $\phi(x_i)$ for every data point x_i , the dualized representation of the prediction (4.3) allows the computation of $\hat{\mu}_{a,t}(x)$ only from objects in the kernel matrix.

Next, we construct the widths of the confidence intervals around the prediction. As for linear bandits we find appropriate widths in terms of the Mahalanobis distance of $\phi(x_{a,t})$ from the matrix Φ_t :

$$\hat{\sigma}_{a,t} \stackrel{\text{def}}{=} \sqrt{\phi(x_{a,t})^\top (\Phi_t^\top \Phi_t + \gamma I)^{-1} \phi(x_{a,t})}. \quad (4.4)$$

Once again we motivate this choice of width by noting that it is exactly the variance of the prediction estimator when the noise in the dualized data is standard normal. In order to compute these widths we derive a dualized representation of (4.4). Our derivation is similar to the kernelization of the Mahalanobis distance for centered data by Haasdonk and Peřalska (2010): Since the matrices $(\Phi_t^\top \Phi_t + \gamma I)$ and $(\Phi_t \Phi_t^\top + \gamma I)$ are regularized, they are strictly positive definite, and therefore

$$\begin{aligned} (\Phi_t^\top \Phi_t + \gamma I) \Phi_t^\top &= \Phi_t^\top (\Phi_t \Phi_t^\top + \gamma I), \\ \Phi_t^\top (\Phi_t \Phi_t^\top + \gamma I)^{-1} &= (\Phi_t^\top \Phi_t + \gamma I)^{-1} \Phi_t^\top. \end{aligned}$$

Now, we can extract the Mahalanobis distance from the last equation

$$(\Phi_t^\top \Phi_t + \gamma I) \phi(x) = (\Phi_t^\top \mathbf{k}_{x,t} + \gamma \phi(x)),$$

from which we deduce that

$$\phi(x) = \Phi_t^\top (\Phi_t \Phi_t^\top + \gamma I)^{-1} \mathbf{k}_{x,t} + \gamma (\Phi_t^\top \Phi_t + \gamma I)^{-1} \phi(x)$$

and express $\phi(x)^\top \phi(x)$ as

$$\mathbf{k}_{x,t}^\top (\Phi_t \Phi_t^\top + \gamma I)^{-1} \mathbf{k}_{x,t} + \gamma \phi(x)^\top (\Phi_t^\top \Phi_t + \gamma I)^{-1} \phi(x).$$

Rearranging, we get an expression for the width involving only inner products,

$$\widehat{\sigma}_{a,t} \stackrel{\text{def}}{=} \gamma^{-1/2} \sqrt{k(x_{a,t}, x_{a,t}) - \mathbf{k}_{x_{a,t},t}^\top (\mathbf{K}_t + \gamma I)^{-1} \mathbf{k}_{x_{a,t},t}}. \quad (4.5)$$

As for `LinUCB`, `KernelUCB` chooses the action a_t at time t which satisfies

$$a_t \stackrel{\text{def}}{=} \arg \max_{a \in A} \left(\mathbf{k}_{x_{a,t},t}^\top (\mathbf{K}_t + \gamma I)^{-1} y_t + \frac{\eta}{\gamma^{1/2}} \sqrt{k(x_{a,t}, x_{a,t}) - \mathbf{k}_{x_{a,t},t}^\top (\mathbf{K}_t + \gamma I)^{-1} \mathbf{k}_{x_{a,t},t}} \right),$$

where η is a (possibly time dependent) exploration parameter of the algorithm. Considering a_t and $\widehat{\sigma}_{a,t}$ we see that `GP-UCB` is a special case of `KernelUCB` where the regularization constant is set to the model noise.

The selection of an appropriate kernel function is problem dependent (Shawe-Taylor and Cristianini 2004). The linear kernel corresponds to $\phi \equiv \text{Id}$ and leads to the dual representation of the `LinUCB` algorithm in the primal. A nonlinear kernel function creates a kernelized UCB algorithm for a nonlinear bandit. Typical examples of nonlinear kernel functions include: the radial basis function where $k(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / 2\sigma^2)$, for $\sigma > 0$ and the polynomial kernel $k(x_i, x_j) = (x_i^\top x_j + 1)^P$.

4.2 Analysis of `KernelUCB`

If we directly applied known regret bounds (Auer 2002; Chu et al. 2011) for linear contextual bandits to our setting, we would obtain a bound in terms of the dimension of the RKHS, which is possibly infinite.

We avoid this problem through a careful consideration of the eigenvalues of the covariance matrix and the choice of the regularisation constant and give a bound in terms of a data dependent quantity \widetilde{d} which we call the *effective dimension*: Let $(\lambda_{i,t})_{i \geq 1}$ denote the eigenvalues of $C_t^\gamma = \Phi_t^\top \Phi_t + \gamma I$ in decreasing order and define

$$\widetilde{d} \stackrel{\text{def}}{=} \min\{j : j\gamma \ln T \geq \Lambda_{T,j}\} \text{ where } \Lambda_{T,j} \stackrel{\text{def}}{=} \sum_{i>j} \lambda_{i,T} - \gamma.$$

We call \widetilde{d} the effective dimension because it gives a proxy for the number of principal directions over which the projection of the data in the RKHS is spread. If the data all fall within a subspace of \mathcal{H} of dimension D' , then $\Lambda_{T,D'} = 0$ and $\widetilde{d} \leq D'$.

However, more generally, \widetilde{d} can be thought of as a measure of how quickly the eigenvalues of $\Phi_t^\top \Phi_t$ are decreasing. For example if the eigenvalues are only polynomially decreasing in i (i.e., $\lambda_i \leq Ci^{-\alpha}$ for some $\alpha > 1$ and some constant $C > 0$) then $\widetilde{d} \leq 1 + (C/(\gamma \ln T))^{1/\alpha}$.

In order to get a better dependence of \widetilde{d} , we analyze a related algorithm, `SupKernelUCB`, that uses the elimination technique¹ of Auer (2002). With `SupKernelUCB`, however, the set of arms can no longer be changing.

¹another option would be an approach similar to `LinearEliminator` of Theorem 1.1.2

Theorem 4.2.1 — Regret of SupKernelUCB by Valko et al. (2013b). Assume that $\|\phi(x_{a,t})\| \leq 1$ and $|r_{a,t}| \in [0, 1]$ for all $a \in A$ and $t \geq 1$, and set $\eta = \sqrt{2 \ln 2TN/\delta}$. Then with probability $1 - \delta$, SupKernelUCB satisfies:

$$R_T \leq \left[2 + 2 \left(1 + \sqrt{\frac{\gamma}{2 \ln(2TN(1 + \ln T)/\delta)}} \right) \|\theta^*\| + \right. \\ \left. + 8 \sqrt{\left(12 + \frac{15}{\gamma} \right) \max \left\{ \ln \left(\frac{T}{\tilde{d}\gamma} + 1 \right), \ln T \right\}^3 \times \sqrt{\left(2 \ln \frac{2TN(1 + \ln T)}{\delta} \right)}} \right] \sqrt{\tilde{d}T}$$

- R When $\Phi \equiv \text{Id}$, $\tilde{d} \leq D$, the assumption that $\|\phi(x_{a,t})\| \leq 1$ becomes the assumption that the contexts are normalised in the primal, and we recover exactly the result of Chu et al. 2011 which matches the lower bound for this setting.
- R Theorem 4.2.1 suggests that if we know that $\|\theta^*\| \leq L$, for some L , we should set γ to be of the order of L^{-1} so that we obtain an $\tilde{\mathcal{O}}(\sqrt{L\tilde{d}T})$ regret. If we do not have such knowledge, just setting γ to a constant (e.g., found by a cross-validation) will incur $\tilde{\mathcal{O}}(\|\theta^*\| \sqrt{\tilde{d}T})$ regret.

4.3 Relationship with GP-UCB

We now relate our analysis to that of GP-UCB by Srinivas et al. (2010), and in particular to their Theorem 3, which treats the agnostic case. In this case, θ^* is not assumed to be sampled from a GP, but instead to have a bounded RKHS norm $\|\theta^*\|$. Under this assumption, the cumulative regret is bounded as

$$\mathcal{O} \left((I(y_T; \theta^*) + \|\theta^*\|^2 \sqrt{I(y_T; \theta^*)}) \sqrt{T} \right), \quad (4.6)$$

where $I(y_T; \theta^*)$ is the mutual information between θ^* and the vector of (noisy) observations y_T . Both $I(y_T; \theta^*)$ in (4.6) and \tilde{d} are data-dependent quantities. We now relate them in order to compare the analyses. We have that:

$$\begin{aligned} I(y_T; f) &= \ln |I + \sigma^{-2} \mathbf{K}_T| = \sum_i \ln(1 + \sigma^{-2} \lambda_{i,T}) \\ &\geq \ln \left(1 + \sigma^{-2} \lambda_{\tilde{d}-1,T} \right) \left(\tilde{d} - 1 + \frac{\sum_{i > \tilde{d}-1} \lambda_{i,T}}{\lambda_{\tilde{d}-1,T}} \right) \\ &\geq (\tilde{d} - 1) \ln \left(1 + \sigma^{-2} \lambda_{\tilde{d}-1,T} \right) \left[1 + \frac{\gamma \ln T}{\lambda_{\tilde{d}-1,T}} \right] \\ &\geq (\tilde{d} - 1) \max_B \min \left\{ \ln(1 + B) \gamma \sigma^{-2} \ln(T), \frac{\ln(1 + B)}{B} \right\} \\ &\geq \Omega(\tilde{d} \ln \ln T) \end{aligned}$$

In the second equality, we used the fact that the eigenvalues of $\Phi_T^\top \Phi_T$ are the same as the eigenvalues of $\Phi_T \Phi_T^\top$. In the second inequality we used the definition of \tilde{d} . For the second to last inequality we considered the two cases when $\lambda_{\tilde{d}-1,T} \leq B\sigma^2$ and when $\lambda_{\tilde{d}-1,T} \geq B\sigma^2$ for some B .

This shows that \tilde{d} is at least as good as $I(y_T; \theta^*)$, and comparing our Theorem 4.2.1 with (4.6), our regret bound only scales as $O(\sqrt{\tilde{d}})$, while the dependence of the regret bound (4.6) is linear in $I(y_T; \theta^*)$. In particular, this means that for the linear kernel we attain the lower bound for linear contextual bandits (Chu et al. 2011) while GP-UCB is \sqrt{D} away. This concerns only the agnostic case of GP-UCB, i.e., Theorem 3 by Srinivas et al. (2010), which is the same setting as ours. When θ^* is sampled from a GP, their result for linear case also matches the lower bound.

Srinivas et al. (2010) also provide an upper bound on $I(y_T; \theta^*)$, denoted by γ_T , for certain kernels. As a consequence of the link between $I(y_T; \theta^*)$, γ_T and \tilde{d} , we may also express our bounds in terms of γ_T . Moreover, in the agnostic case again, our bounds enjoy an improved dependence on this parameter: for example, for the widely used RBF kernel, our bound scales with $O(\ln T)^{D/2}$ in place of $O(\ln T)^D$.

Finally, when $\|\theta^*\|$ is unknown and we are unable to regularize appropriately, our regret bound only depends on $\|\theta^*\|$ linearly (Remark 4.2), while the dependence in (4.6) is quadratic.

4.4 Perspectives of bandits for stochastic processes

In this chapter, we worked with finite (discrete) action spaces. However, Gaussian processes (GPs) define a distribution of (continuous) *functions* where the smoothness properties are governed by a covariance function (kernel) \mathbf{K} . Therefore, a natural extension of the setting considered in this chapter is an optimization of a continuous function (on a bounded domain), with either a bounded RKHS norm (in a frequentist case) or sampled from a GP. A clear candidate, especially in the GP case is ThompsonSampling. Since the sample, in this case, is a function, the maximization is not trivial in general. One option is to sequentially discretize the domain of the given function, for example as done by Contal and Vayatis (2016) using upper confidence bounds. This approach is related to general *black-box function optimization* that we discuss in Chapter 6. Furthermore, it may be possible to extend the discretization techniques to other stochastic processes, for example to *Brownian motion*.

On a practical side, kernel and GP bandits are based on *kernel ridge regression* (KRR) (Schölkopf and Smola 2001; Shawe-Taylor and Cristianini 2004) which comes with computational challenges. The kernel matrix grows and so does the per-step computation time, which is undesirable in any sequential setting. This problem is not specific to bandits and appears in online kernel regression or online PCA as well. A typical solution in the offline or batch case is the *Nyström family* of algorithms which randomly selects a subset of columns from the kernel matrix that is used to construct a low-rank approximation. The quality of the approximate solution is strongly affected by the sampling distribution and the number of columns selected (Rudi et al. 2015). For KRR, Alaoui and Mahoney (2015) introduce a concept of *ridge leverage scores* (RLSs) of a square matrix, and shows that Nyström approximations sampled according to RLS have strong reconstruction guarantees that translate into good guarantees for the approximate KRR solution (Alaoui and Mahoney 2015; Rudi et al. 2015). We can apply the Nyström method with RLSs for the online setting (Calandriello et al. 2016) and in particular to the bandit case where the kernel matrix being constructed online may have a specific behavior: Notice that in the cumulative regret optimization, the (well performing) algorithms would choose more and more near-optimal points (arms) and therefore the data from which we construct the kernel matrix are more and more *correlated*. This specific behavior could be in turn used for more adaptive and space-saving approximation of the kernel matrix.

5. Polymatroid bandits

In this chapter, we first introduce polymatroids and illustrate them on practical problems. We use the problem of the *minimum-cost flow* (Megiddo 1974) on a network as an illustrative example before we give the formal definition of polymatroids and learning with them.

■ **Example 5.1** Consider a flow network with L source nodes and one sink node. The network is illustrated in Figure 5.1.

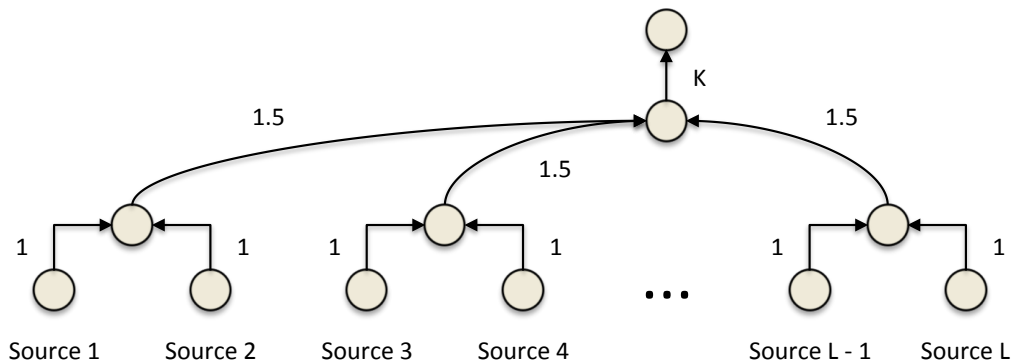


Figure 5.1: The flow network contains L source nodes and the maximum flow is K . The capacity of the link is shown next to the link.

The network is defined by three constraints. First, the maximum flow through any source node is 1. Second, the maximum flow through any two consecutive source nodes, e and $e + 1$ where $e = 2i - 1$ for $i \in \{1, \dots, L/2\}$, is $\frac{3}{2}$. Third, the maximum flow is K . We assume that K is an integer multiple of $\frac{3}{2}$. The cost of the flow from source node e is a Bernoulli random variable with mean:

$$\bar{w}(e) = \begin{cases} 0.5 - \Delta/2 & e \leq \frac{4}{3}K \\ 0.5 + \Delta/2 & \text{otherwise.} \end{cases} \quad (5.1)$$

Our problem is parametrized by K , L , and Δ . The optimal solution to the problem is to pass the maximum flow through the first $\frac{4}{3}K$ source nodes.

Our problem can be formulated as minimizing a *modular* function on a polymatroid. The ground set E are L source nodes. The *submodular* function f captures the structure of the network and is defined as

$$f(X) = \min \left\{ \sum_{i=1}^{L/2} \min \left\{ \mathbb{1}\{(2i-1) \in X\} + \mathbb{1}\{2i \in X\}, \frac{3}{2} \right\}, K \right\}. \quad (5.2)$$

Note that $f(X)$ can be computed in $\mathcal{O}(L)$ time, by summing up L indicators. The *weight* of item e is drawn i.i.d. from a Bernoulli distribution with mean $\bar{\mathbf{w}}(e)$ in (5.1), independently of the other items. ■

With this example in mind we formalize the notion of a polymatroid. A *polymatroid* (Edmonds 1970) is a polytope associated with a submodular function. More specifically, a polymatroid is a pair $M = (E, f)$. In this definition, $E = \{1, \dots, L\}$ is a *ground set* of L items. In our flow problem, E is the set of L *sources* of the flow network. Furthermore $f : 2^E \rightarrow \mathbb{R}^+$ is a function from the power set of E to non-negative real numbers. The function f is *monotonic*, $\forall X \subseteq Y \subseteq E : f(X) \leq f(Y)$; *submodular*, $\forall X, Y \subseteq E : f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$; and $f(\emptyset) = 0$. In the flow problem, $f(X)$ is the maximum flow through source nodes $X \subseteq E$. Since f is monotonic, $f(E)$ is one of its maxima. We refer to $f(E)$ as the *rank* of a polymatroid and denote it by K . For the flow problem, K is the value of the maximum flow. Without loss of generality, we assume that $f(e) \leq 1$ for all items $e \in E$. Because f is submodular, we indirectly assume that $f(X+e) - f(X) \leq 1$ for all $X \subseteq E$. In the flow problem, this constraint translates to assuming that the value of any source in the network is upper bounded by one. The *independence polyhedron* P_M associated with polymatroid M is a compact subset of \mathbb{R}^L defined as

$$P_M = \{ \mathbf{x} : \mathbf{x} \in \mathbb{R}^L, \mathbf{x} \geq 0, \forall X \subseteq E : \sum_{e \in X} \mathbf{x}(e) \leq f(X) \}, \quad (5.3)$$

where $\mathbf{x}(e)$ is the e -th entry of vector \mathbf{x} . The vector \mathbf{x} is *independent* if $\mathbf{x} \in P_M$. In the flow example, $\mathbf{x}(e) \leq 1$ denotes how much of the unit flow goes through the source e and P_M is the set of all possible flows respecting the constraints of a given network. The *base polyhedron* B_M is a subset of P_M defined as

$$B_M = \{ \mathbf{x} : \mathbf{x} \in P_M, \sum_{e \in E} \mathbf{x}(e) = K \}. \quad (5.4)$$

The vector \mathbf{x} is a *basis* if $\mathbf{x} \in B_M$. In other words, \mathbf{x} is independent and its entries sum up to K . For the flow network B_M is the set of all possible *maximum* flows.

5.1 Optimization on polymatroids

A *weighted polymatroid* is a polymatroid associated with a vector of weights $\mathbf{w} \in (\mathbb{R}^+)^L$. The e -th entry of \mathbf{w} , $\mathbf{w}(e)$, is the weight of item e . For instance $\mathbf{w}(e)$ can be the cost of a unit flow going through source e and for a particular flow \mathbf{x} , $\langle \mathbf{w}, \mathbf{x} \rangle$ is the value of the flow given a weight assignment \mathbf{w} . A classic problem in polyhedral optimization is to find a *maximum-weight basis* of a polymatroid,

$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in B_M} \langle \mathbf{w}, \mathbf{x} \rangle = \arg \max_{\mathbf{x} \in P_M} \langle \mathbf{w}, \mathbf{x} \rangle. \quad (5.5)$$

This basis can be computed greedily (Algorithm 2). The greedy algorithm works as follows. First, the items E are sorted in decreasing order of their weights, $\mathbf{w}(e_1) \geq \dots \geq \mathbf{w}(e_L)$. We assume that the ties are broken by an arbitrary but fixed rule. Second, \mathbf{x}^* is computed as $\mathbf{x}^*(e_i) = f(\{e_1, \dots, e_i\}) -$

Algorithm 2 Greedy: Edmond’s algorithm for the maximum-weight basis of a polymatroid.

Input: Polymatroid $M = (E, f)$, weights \mathbf{w}
 Let e_1, \dots, e_L be an ordering of items such that:
 $\mathbf{w}(e_1) \geq \dots \geq \mathbf{w}(e_L)$
 $\mathbf{x} \leftarrow$ All-zeros vector of length L
for all $i = 1, \dots, L$ **do**
 $\mathbf{x}(e_i) \leftarrow f(\{e_1, \dots, e_i\}) - f(\{e_1, \dots, e_{i-1}\})$
end for
Output: Maximum-weight basis \mathbf{x}

$f(\{e_1, \dots, e_{i-1}\})$ for all i . Note that the *minimum-weight basis* of a polymatroid with weights \mathbf{w} is the maximum-weight basis of the same polymatroid with weights $\max_{e \in E} \mathbf{w}(e) - \mathbf{w}$,

$$\arg \min_{\mathbf{x} \in B_M} \langle \mathbf{w}, \mathbf{x} \rangle = \arg \max_{\mathbf{x} \in B_M} \langle \max_{e \in E} \mathbf{w}(e) - \mathbf{w}, \mathbf{x} \rangle. \quad (5.6)$$

Therefore, the minimization problem is mathematically equivalent to the maximization problem (5.5), and all results in this chapter straightforwardly generalize to the minimization. For instance, the minimum-weight basis of the polymatroid corresponding to a flow network is the maximum flow with the minimum cost (Fujishige 2005), which we refer to as the minimum-cost flow (see Example 5.1).

Many existing problems can be viewed as an optimization on a polymatroid (5.5). For instance, polymatroids generalize *matroids* (Whitney 1935), a notion of independence in combinatorial optimization that is closely related to computational efficiency. In particular, let $M = (E, \mathcal{I})$ be a matroid, where $E = \{1, \dots, L\}$ is its ground set, $\mathcal{I} \subseteq 2^E$ are its independent sets, and

$$f(X) = \max_{Y: Y \subseteq X, Y \in \mathcal{I}} |Y| \quad (5.7)$$

is its *rank function*. Let $\mathbf{w} \in (\mathbb{R}^+)^L$ be a vector of non-negative weights. Then the maximum-weight basis of a matroid,

$$A^* = \arg \max_{A \in \mathcal{I}} \sum_{e \in A} \mathbf{w}(e), \quad (5.8)$$

can be also defined as $A^* = \{e : \mathbf{x}^*(e) = 1\}$, where \mathbf{x}^* is the maximum-weight basis of the corresponding polymatroid. The basis is $\mathbf{x}^* \in \{0, 1\}^L$ because the rank function is a monotonic submodular function with zero-one increments (Fujishige 2005). Our optimization problem can be written as a *linear program* (LP, Bertsimas and Tsitsiklis 1997),

$$\max_{\mathbf{x}} \sum_{e \in E} \mathbf{w}(e) \mathbf{x}(e), \quad \text{s.t.:} \quad \sum_{e \in X} \mathbf{x}(e) \leq f(X) \quad \forall X \subseteq E, \quad (5.9)$$

where $\mathbf{x} \in (\mathbb{R}^+)^L$ is a vector of L optimized variables. This LP has exponentially many constraints, one for each subset $X \subseteq E$. Therefore, it cannot be solved directly. Nevertheless, Greedy can solve the problem in $O(L \log L)$ time. Therefore, our problem is a very efficient form of linear programming.

The problem of recommending diverse items can be also cast as an optimization on a polymatroid (Ashkan et al. 2014; Ashkan et al. 2015). Let E be a set of recommendable items, $f(X)$ be the number of topics covered by items X , and \mathbf{w} be a weight vector such that $\mathbf{w}(e)$ is the popularity of item e . Then $\mathbf{x}^* = \text{Greedy}(M, \mathbf{w})$ is a vector such that $\mathbf{x}^*(e) > 0$ if and only if item e is the most popular item in at least one topic covered by item e . We illustrate this concept with a simple example.

■ **Example 5.2** Let the ground set E be a set of 3 movies:

e	Movie title	Popularity $w(e)$	Movie genres
1	Inception	0.8	Action
2	Grown Ups 2	0.5	Comedy
3	Kindergarten Cop	0.6	Action Comedy

Let $f(X)$ be the number of movie genres covered by movies X . Then f is submodular and defined as:

$$\begin{aligned} f(\emptyset) &= 0, & f(\{2\}) &= 1, & f(\{1,2\}) &= 2, & f(\{2,3\}) &= 2, \\ f(\{1\}) &= 1, & f(\{3\}) &= 2, & f(\{1,3\}) &= 2, & f(\{1,2,3\}) &= 2. \end{aligned} \quad (5.10)$$

The maximum-weight basis of polymatroid $M = (E, f)$ is $\mathbf{x}^* = (1, 0, 1)$, and $\{e : \mathbf{x}^*(e) > 0\} = \{1, 3\}$ is the minimal set of movies that cover each movie genre by the most popular movie in that genre. ■

5.2 Combinatorial optimization on polymatroids

In this chapter, we restrict our attention to the feasible solutions,

$$\Theta = \{\mathbf{x} : (\exists \mathbf{w} \in (\mathbb{R}^+)^L : \mathbf{x} = \text{Greedy}(M, \mathbf{w}))\}, \quad (5.11)$$

that can be computed greedily for some weight vector \mathbf{w} and define our objective as finding

$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in \Theta} \langle \mathbf{w}, \mathbf{x} \rangle. \quad (5.12)$$

The set Θ are the vertices of B_M (Kveton et al. 2014). Our choice is motivated by three reasons. First, we study the problem of learning to act greedily. Therefore, we are only interested in the bases that can be computed greedily. Second, many optimization problems of our interest (e.g., recommendation of *diverse items*) are combinatorial in nature and only the bases in Θ are suitable feasible solutions. For instance, in a graphic matroid, Θ is a set of spanning trees. In a linear matroid, Θ is a set of maximal sets of linearly independent vectors. The bases in $B_M \setminus \Theta$ do not have this interpretation. Another example is our recommendations problem in Section 5.1. In this problem, for any $\mathbf{x} = \text{Greedy}(M, \mathbf{w})$, $\{e : \mathbf{x}(e) > 0\}$ is a minimal set of items that cover each topic by the most popular item according to \mathbf{w} . The bases in $B_M \setminus \Theta$ cannot be interpreted in this way. Finally, we note that our choice does not have any impact on the notion of optimality. In particular, let \mathbf{x} be optimal for some \mathbf{w} . Then $\mathbf{x}^g = \text{Greedy}(M, \mathbf{w})$ is also optimal and since $\mathbf{x}^g \in \Theta$, it follows that

$$\max_{\mathbf{x} \in B_M} \langle \mathbf{w}, \mathbf{x} \rangle = \max_{\mathbf{x} \in \Theta} \langle \mathbf{w}, \mathbf{x} \rangle. \quad (5.13)$$

5.3 Learning model

We formalize our learning problem as a polymatroid semi-bandit. A *polymatroid semi-bandit* is a pair (M, \mathcal{P}) , where M is a polymatroid and \mathcal{P} is a probability distribution over the weights $\mathbf{w} \in \mathbb{R}^L$ of items E in M . The e -th entry of \mathbf{w} , $w(e)$, is the weight of item e . We assume that the weights \mathbf{w} are drawn i.i.d. from \mathcal{P} and that \mathcal{P} is unknown. Without loss of generality, we assume that \mathcal{P} is a distribution over the unit cube $[0, 1]^L$. Other than that, we do not assume anything about \mathcal{P} . We denote the expected weights of the items by $\bar{\mathbf{w}} = \mathbb{E}[\mathbf{w}]$. By our assumptions on \mathcal{P} , $\bar{\mathbf{w}}(e) \geq 0$ for all items e . Each item e is associated with an *arm* and each feasible solution

Algorithm 3 OPM: Optimistic polymatroid maximization.

Input: Polymatroid $M = (E, f)$
 Observe $\mathbf{w}_0 \sim \mathcal{P}$ {Initialization}
 $\widehat{\mathbf{w}}_1(e) \leftarrow \mathbf{w}_0(e), \forall e \in E$
 $T_0(e) \leftarrow 1, \forall e \in E$
for all $t = 1, \dots, T$ **do**
 $U_t(e) \leftarrow \widehat{\mathbf{w}}_{T_{t-1}(e)}(e) + c_{t-1, T_{t-1}(e)}, \forall e \in E$ {Compute UCBs}
 $\mathbf{x}_t \leftarrow \text{Greedy}(M, U_t)$ {Find a maximum-weight basis}
 Observe $\{(e, \mathbf{w}_t(e)) : \mathbf{x}_t(e) > 0\}$, where $\mathbf{w}_t \sim P$ {Choose the basis}
 $T_t(e) \leftarrow T_{t-1}(e), \forall e \in E$ {Update statistics}
 $T_t(e) \leftarrow T_t(e) + 1, \forall e : \mathbf{x}_t(e) > 0$
 $\widehat{\mathbf{w}}_{T_t(e)}(e) \leftarrow \frac{T_{t-1}(e)\widehat{\mathbf{w}}_{T_{t-1}(e)}(e) + \mathbf{w}_t(e)}{T_t(e)}, \forall e : \mathbf{x}_t(e) > 0$
end for

$\mathbf{x} \in \Theta$ is associated with a set of arms $A = \{e : \mathbf{x}(e) > 0\}$. The arms A are the items with nonzero contributions in \mathbf{x} . After the arms are *pulled*, the learning agent receives a *payoff* of $\langle \mathbf{w}, \mathbf{x} \rangle$ and *observes* $\{(e, \mathbf{w}(e)) : \mathbf{x}(e) > 0\}$, the weights of all items with nonzero contributions in \mathbf{x} . This feedback model is known as *semi-bandit* (Audibert et al. 2014). The solution to our problem is a maximum-weight basis in expectation,

$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in \Theta} \mathbb{E}_{\mathbf{w}} [\langle \mathbf{w}, \mathbf{x} \rangle] = \arg \max_{\mathbf{x} \in \Theta} \langle \bar{\mathbf{w}}, \mathbf{x} \rangle. \quad (5.14)$$

This problem is equivalent to problem (5.12) and so can be solved greedily, $\mathbf{x}^* = \text{Greedy}(M, \bar{\mathbf{w}})$.

We choose our observation model for several reasons. First, the model is a natural generalization of that in matroid bandits (Kveton et al. 2014). In matroid bandits, the bases are of the form $\mathbf{x} \in \{0, 1\}^L$ and the learning agent observes the weights of all chosen items e , $\mathbf{x}(e) = 1$. In this case, $\mathbf{x}(e) = 1$ is equivalent to $\mathbf{x}(e) > 0$. Second, our observation model is suitable for our motivating examples (Section 5.1). Specifically, in the minimum-cost flow problem, we assume that the learning agent observes the costs of all source nodes that contribute to the maximum flow. In the movie recommendation problem, the agent observes individual movies chosen by the user, from a set of recommended movies. Finally, our observation model allows us to derive similar regret bounds to those in matroid bandits (Kveton et al. 2014).

Our learning problem is *episodic*. Let $(\mathbf{w}_t)_{t=1}^T$ be an i.i.d. sequence of weights drawn from distribution \mathcal{P} . In episode t , the learning agent chooses basis \mathbf{x}_t based on its prior actions $\mathbf{x}_1, \dots, \mathbf{x}_{t-1}$ and observations of $\mathbf{w}_1, \dots, \mathbf{w}_{t-1}$; gains $\langle \mathbf{w}_t, \mathbf{x}_t \rangle$; and observes $\{(e, \mathbf{w}_t(e)) : \mathbf{x}_t(e) > 0\}$, the weights of all items with nonzero contributions in \mathbf{x}_t . The agent interacts with the environment in T episodes. The goal of the agent is to maximize its expected cumulative return, or equivalently to minimize its *expected cumulative regret*,

$$R_T = \mathbb{E}_{\mathbf{w}_1, \dots, \mathbf{w}_T} \left[\sum_{t=1}^T R(\mathbf{x}_t, \mathbf{w}_t) \right], \quad (5.15)$$

where $R(\mathbf{x}, \mathbf{w}) = \langle \mathbf{w}, \mathbf{x}^* \rangle - \langle \mathbf{w}, \mathbf{x} \rangle$ is the regret associated with basis \mathbf{x} and weights \mathbf{w} .

5.4 The OPM algorithm

Our learning algorithm is designed based on the *optimism in the face of uncertainty* principle (Auer et al. 2002a). In particular, it is a greedy method for finding a maximum-weight basis

of a polymatroid where the expected weight $\bar{\mathbf{w}}(e)$ of each item is substituted with its optimistic estimate $U_t(e)$. We refer to our method as *Optimistic Polymatroid Maximization (OPM)*.

The pseudocode of **OPM** is given in Algorithm 3. In each episode t , the algorithm works as follows. First, we compute an *upper confidence bound* (UCB) on the expected weight of each item e ,

$$U_t(e) = \widehat{\mathbf{w}}_{T_{t-1}(e)}(e) + c_{t-1, T_{t-1}(e)}, \quad (5.16)$$

where $\widehat{\mathbf{w}}_{T_{t-1}(e)}(e)$ is our estimate of the expected weight $\bar{\mathbf{w}}(e)$ in episode t , $c_{t-1, T_{t-1}(e)}$ is the radius of the confidence interval around this estimate, and $T_{t-1}(e)$ denotes the number of times that item e is selected in the first $t-1$ episodes, $\mathbf{x}_i(e) > 0$ for $i < t$. Second, we compute the maximum-weight basis with respect to U_t using Greedy. Finally, we select the basis, observe the weights of all items e where $\mathbf{x}_t(e) > 0$, and then update our model $\widehat{\mathbf{w}}$ of the environment. The radius

$$c_{t,s} = \sqrt{\frac{2 \log t}{s}} \quad (5.17)$$

is designed such that each UCB is a high-probability upper bound on the corresponding weight $\widehat{\mathbf{w}}_s(e)$. The UCBs encourage exploration of items that have not been observed sufficiently often. As the number of past episodes increases, we get better estimates of the weights $\bar{\mathbf{w}}$, all confidence intervals shrink, and **OPM** starts exploiting most rewarding items. The $\log(t)$ term increases with time and enforces continuous exploration.

For simplicity of exposition, we assume that **OPM** is initialized by observing each item once. In practice, this initialization step can be implemented efficiently in the first L episodes. In particular, in episode $t \leq L$, **OPM** chooses first item t and then all other items, in an arbitrary order. The corresponding regret is bounded by KL because $\langle \bar{\mathbf{w}}, \mathbf{x} \rangle \in [0, K]$ for any $\bar{\mathbf{w}}$ (Section 5.3) and basis \mathbf{x} .

OPM is a greedy method and therefore is extremely computationally efficient. In particular, suppose that the function f is an oracle that can be queried in $\mathcal{O}(1)$ time. Then the time complexity of **OPM** in episode t is $\mathcal{O}(L \log L)$, comparable to that of sorting L numbers. The design of **OPM** is not very surprising and it draws on prior work (Gai et al. 2012; Kveton et al. 2014).

Our major contribution is that we derive a tight upper bound on the regret of **OPM**. Our analysis is a significant improvement over the one of Kveton et al. (2014), who analyze the regret of **OPM** in the context of matroids. Roughly speaking, the analysis of Kveton et al. (2014) leverages the augmentation property of a matroid. Our analysis is based on the submodularity of a polymatroid and we state the distribution independent (gap-free) regret bound below.

Theorem 5.4.1 — Regret of OPM by Kveton et al. 2016. In any stochastic polymatroid semi-bandit, the regret of **OPM** is bounded as:

$$R_T \leq 8\sqrt{KLT \log T} + \frac{4}{3}\pi^2 L^2.$$

5.5 Discussion and perspectives of polymatroid bandits

The bound of Theorem 5.4.1 is at most linear in K and L , and sublinear in T . In other words, it scales favorably with all quantities of interest and therefore we expect it to be practical. Our $\mathcal{O}(\sqrt{KLT \log T})$ upper bound matches the following lower bound up to a factor of $\sqrt{\log T}$, which is a corollary of Theorem 0.0.1 (Auer et al. 2002b).

Corollary 5.5.1 — Lower bound for matroid bandits by Kveton et al. 2016. For any L and K such that L/K is an integer, and any $T > 0$, the regret of any algorithm on a partition matroid bandit is bounded from below as

$$R_T \geq \frac{1}{20} \min(\sqrt{KLT}, KT).$$

Notice that the stated lower bound is for matroids. However, it is an open question whether the factor of L in polymatroids is inherent. It is possible that learning in polymatroids is harder than in matroids (where the factor is $L - K$) because the order in which the learning algorithm chooses optimal items matters.

In this chapter, we studied one particular problem, the maximization of a modular function on a polymatroid, in one particular learning setting, stochastic semi-bandits. It is an open question whether the ideas in our paper generalize to other polymatroid problems, such as maximizing a modular function on the intersection of two matroids (Papadimitriou and Steiglitz 1998); and other learning variants of our problem, such as learning in the adversarial setting (Auer et al. 2002b) or with the full-bandit feedback. Several recent papers studied the problem of learning how to maximize a submodular function (Gabillon et al. 2013; Gabillon et al. 2014; Guillory and Bilmes 2011; Wen et al. 2013; Yue and Guestrin 2011). These are only loosely related to this work because they study a different problem, which is learning how to maximize an *unknown submodular function* subject to a cardinality constraint. Our learning problem is maximizing an *unknown modular function* subject to a *known submodular constraint*.

6. Bandits for function optimization

In this chapter, we apply bandit approaches to the problem of optimizing a function $f : \mathcal{X} \rightarrow \mathbb{R}$ given a finite budget of T noisy evaluations. We consider that the cost of any of these *function evaluations* is high. That means we care about assessing the optimization performance in terms of the sample complexity, i.e., the number of T function evaluations. This is typically the case when one needs to tune parameters for a complex system seen as a *black-box*, which performance can only be evaluated by a costly simulation. One such example is the *hyper-parameter tuning* where the sensitivity to perturbations is large and the derivatives of the objective function with respect to these parameters do not exist or are unknown.

Such setting is another instance of the sequential decision-making setting under *bandit feedback*. In this setting, the actions are the points that lie in a domain \mathcal{X} . At each step t , an algorithm selects an action $x_t \in \mathcal{X}$ and receives a reward r_t , which is a noisy function evaluation such that $r_t = f(x_t) + \varepsilon_t$, where ε_t is a bounded noise with $\mathbb{E}[\varepsilon_t | x_t] = 0$. After T evaluations, the algorithm outputs its best guess $x(T)$, which can be different from x_T . The performance measure we want to minimize is the value of the function at the returned point compared to the optimum, also referred to as *simple regret*,

$$R_T \stackrel{\text{def}}{=} \sup_{x \in \mathcal{X}} f(x) - f(x(T)).$$

We assume there exists at least one point $x^* \in \mathcal{X}$ such that $f(x^*) = \sup_{x \in \mathcal{X}} f(x)$. The relationship with bandit settings motivated UCT (Coquelin and Munos 2007; Kocsis and Szepesvári 2006), an empirically successful heuristic (Coulom 2007; Gelly et al. 2006; Silver et al. 2016) that hierarchically partitions domain \mathcal{X} and selects the next point $x_t \in \mathcal{X}$ using upper confidence bounds (Auer et al. 2002a). The empirical success of UCT on one side but the absence of performance guarantees for it on the other, incited research on similar but theoretically founded algorithms (Azar et al. 2014; Bubeck et al. 2011b; Bull 2015; Grill et al. 2015; Kleinberg et al. 2008; Munos 2014).

As the global optimization of the unknown function without absolutely any assumptions would be a daunting needle-in-a-haystack problem, most of the algorithms assume at least a very weak assumption that the function does *not decrease faster than a known rate* around *one* of its global optima. In other words, they assume a certain *local smoothness* property of f . This smoothness is

often expressed in the form of a semi-metric ℓ that quantifies this regularity (Bubeck et al. 2011b). Naturally, this regularity also influences the guarantees that these algorithms are able to furnish. Many of them define a *near-optimality dimension* d or a *zooming dimension*. These are ℓ -dependent quantities used to bound the simple regret R_T or a related notion called *cumulative regret*. Table 6.1 lists some of the algorithms for the setting with both *known* and *unknown* smoothness of f ; and for both *stochastic* and *deterministic* (where $\varepsilon_t = 0$ for all t) function evaluations. In the rest of the chapter, we focus on the stochastic case.

	deterministic	stochastic
known smoothness	DOO	Zooming, HOO, HCT
unknown smoothness	DiRect, SOO	StoS00, TaxonomyZoom, ATB, POO

Table 6.1: Hierarchical optimistic optimization algorithms

6.1 Near-optimality dimension independent of a semi-metric

In our recent work (Grill et al. 2015) we gave a notion of such near-optimality dimension d that does not directly relate the smoothness property of f to a specific metric ℓ but *directly* to the *hierarchical partitioning* $\mathcal{P} = \{\mathcal{P}_{h,i}\}$, a *tree-based representation* of the space used by the algorithm. Indeed, an interesting fundamental question is to determine a good characterization of the difficulty of the optimization for an algorithm that uses a given hierarchical partitioning of space \mathcal{X} as its input. The kind of hierarchical partitioning $\{\mathcal{P}_{h,i}\}$ we consider is similar to the ones introduced in prior work: for any depth $h \geq 0$ in the tree representation, the set of *cells* $\{\mathcal{P}_{h,i}\}_{1 \leq i \leq I_h}$ form a partition of \mathcal{X} , where I_h is the number of cells at depth h . At depth 0, the root of the tree, there is a single cell $\mathcal{P}_{0,1} = \mathcal{X}$. A cell $\mathcal{P}_{h,i}$ of depth h is split into several children subcells $\{\mathcal{P}_{h+1,j}\}_j$ of depth $h+1$. We refer to the standard partitioning (Figure 6.1) as to one where each cell is split into regular same-sized subcells (Preux et al. 2014).

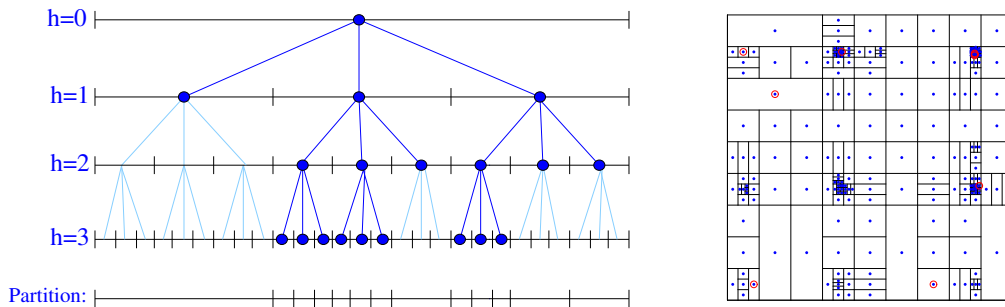


Figure 6.1: Standard partitioning in one dimension (**left**) and two dimensions (**right**).

An important insight (Grill et al. 2015, Section 2) is that a near-optimality dimension d that is independent from the partitioning used by an algorithm (as defined in prior work Azar et al. 2014; Bubeck et al. 2011b; Kleinberg et al. 2008) *does not embody the optimization difficulty perfectly*. This is easy to see, as for any f we could define a partitioning, perfectly suited for f . An example is a partitioning, that at the root splits \mathcal{X} into $\{x^*\}$ and $\mathcal{X} \setminus x^*$, which makes the optimization trivial, whatever d is. This insight was already observed by Slivkins (2011) and Bull (2015), whose *zooming dimension* depends both on the function and the partitioning.

Therefore, we defined (Grill et al. 2015) a notion of near-optimality dimension d which measures the complexity of the optimization problem *directly in terms of the partitioning* used by an algorithm. First, we make the following local smoothness assumption about the function, expressed in terms of the partitioning and *not any metric*: For a given partitioning \mathcal{P} , we assume that there exist $v > 0$ and $\rho \in (0, 1)$, s.t.,

$$\forall h \geq 0, \forall x \in \mathcal{P}_{h, i_h^*}, \quad f(x) \geq f(x^*) - v\rho^h,$$

where (h, i_h^*) is the (unique) cell of depth h containing x^* . Then, we define the near-optimality dimension $d(v, \rho)$ as

$$d(v, \rho) \stackrel{\text{def}}{=} \inf \left\{ d' \in \mathbb{R}^+ : \exists C > 0, \forall h \geq 0, \mathcal{N}_h(2v\rho^h) \leq C\rho^{-d'h} \right\},$$

where for all $\varepsilon > 0$, $\mathcal{N}_h(\varepsilon)$ is the number of cells $\mathcal{P}_{h,i}$ of depth h s.t. $\sup_{x \in \mathcal{P}_{h,i}} f(x) \geq f(x^*) - \varepsilon$. Intuitively, functions with smaller d are easier to optimize and we denote (v, ρ) , for which $d(v, \rho)$ is the smallest, as (v_*, ρ_*) . Obviously, $d(v, \rho)$ depends on \mathcal{P} and f , but *does not depend* on any choice of a specific metric. This definition of d ¹ encompasses the optimization complexity *better* and it is not an artifact of our analysis since many algorithms, such as H00 (Bubeck et al. 2011b), Zooming (Slivkins 2011), StoS00 (Valko et al. 2013a), or HCT (Azar et al. 2014), can be shown to scale with this notion of d . An example of a function with nonzero d is in Figure 6.2.

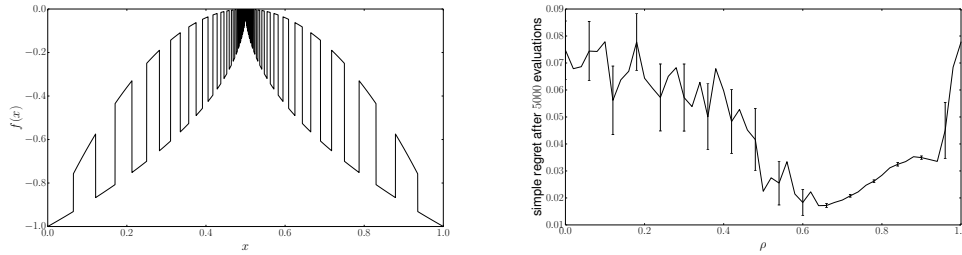


Figure 6.2: Difficult function $f : x \rightarrow s(\log_2 |x - 0.5|) \cdot (\sqrt{|x - 0.5|} - (x - 0.5)^2) - \sqrt{|x - 0.5|}$ where, $s(x) = 1$ if the fractional part of x , that is, $x - \lfloor x \rfloor$, is in $[0, 0.5]$ and $s(x) = 0$, if it is in $(0.5, 1)$. *Left*: Oscillation between two envelopes of different smoothness leading to a nonzero d for a standard partitioning. *Right*: Regret of H00 after 5000 evaluations for different values of ρ .

6.2 Hierarchical optimistic optimization: H00

One of the first known algorithms applying bandit approach to function optimization is H00 (Bubeck et al. 2011b), which assumed the knowledge of the function smoothness. H00 follows an optimistic strategy close to UCT (Kocsis and Szepesvári 2006), but unlike UCT, it uses proper confidence bounds to provide theoretical guarantees. H00 refines a partition of the space based on a hierarchical partitioning, where at each step, a yet unexplored cell (a leaf of the corresponding tree) is selected, and the function is evaluated at a point within this cell. The selected path (from the root to the leaf) is the one that maximizes the minimum value $U_{h,i}(t)$ among all cells of each depth, where the value $U_{h,i}(t)$ of any cell $\mathcal{P}_{h,i}$ is defined as

$$U_{h,i}(t) = \hat{\mu}_{h,i}(t) + \sqrt{\frac{2\ln(t)}{N_{h,i}(t)}} + v\rho^h,$$

¹we use the simplified notation d instead of $d(v, \rho)$ for clarity when no confusion is possible

where t is the number of evaluations done so far, $\widehat{\mu}_{h,i}(t)$ is the empirical average of all evaluations done within $\mathcal{P}_{h,i}$, and $N_{h,i}(t)$ is the number of them. The second term in the definition of $U_{h,i}(t)$ is a Chernoff-Hoeffding type confidence interval, measuring the estimation error induced by the noise. The third term, $v\rho^h$ with $\rho \in (0, 1)$ is, by assumption, a bound on the difference $f(x^*) - f(x)$ for any $x \in \mathcal{P}_{h,i}$, a cell containing x^* . It is this bound, where H00 relies on the knowledge of the smoothness, because the algorithm requires the values of v and ρ . As a consequence of the analysis of H00 (Bubeck et al. 2011a; Bubeck et al. 2011b) using the assumption from Section 6.1 the simple regret of H00 can be bounded as follows.

Theorem 6.2.1 — Simple regret of H00 by Bubeck et al. 2011a. Let R_T be the simple regret of H00 at step T . Let $d(v_*, \rho_*)$ be the near-optimality dimension verifying the assumption from Section 6.1 there exists κ such that for all T , then for any $d' \geq d(v_*, \rho_*)$

$$\mathbb{E}[R_T] \leq \kappa ((\ln T)/T)^{1/(d'+2)}.$$

H00 was later followed by HCT (Azar et al. 2014) that needs to assume a slightly stronger condition on the cell and has a better dependency on the smoothness. However, as H00, also HCT and other algorithms assume that the smoothness of the optimized function is *known*. This is the case of known *semi-metric* (Azar et al. 2014; Bubeck et al. 2011b) and *pseudo-metric* (Kleinberg et al. 2008). This assumption limits the application of these algorithms and opened a very compelling question of whether this knowledge is necessary. We provide some answers in the following.

6.3 Unknown function smoothness: StoS00

We now describe **StoS00**, an algorithm for stochastic function evaluations that does not require the knowledge of the smoothness. **StoS00** operates in the traversals of the tree \mathcal{T} , starting from the root down to the current depth, that is upper bounded by h_{\max} , a parameter of the algorithm. During each traversal, **StoS00** selects a set of promising nodes, at most one per depth h . These nodes are then either *evaluated* or *expanded*.

Evaluating a node at time t means sampling the function in the representative point $x_{h,i}$ of the cell $\mathcal{X}_{h,i}$ and observing the evaluation r_t . Expanding a node (h, i) , means splitting its corresponding cell into its K subcells corresponding to the children:

$$\{(h+1, i_1), (h+1, i_2), \dots, (h+1, i_K)\}.$$

We denote by \mathcal{L} the set of leaves in \mathcal{T} , i.e., the nodes with no children. At any time, only the leaves are eligible for an evaluation or expansion and we never expand the leaves beyond depth h_{\max} . If the function f were deterministic, such as in S00 (Munos 2011), we would expand (simultaneously) any leaf (h, i) whose value $f(x_{h,i})$ is the largest among all leaves of the same or a lower depth, because all such nodes may contain x^* . Unfortunately, we do not receive $f(x_{h,i})$, but only a noisy estimate r_t . Therefore, the main algorithmic idea of **StoS00** is to evaluate the leaves several times in order to build a confident estimate of $f(x_{h,i})$. For this purpose, let us define $\widehat{\mu}_{h,i}(t) = \frac{1}{T_{h,i}(t)} \sum_{s=1}^t r_s \mathbb{1}\{x_s \in \mathcal{X}_{h,i}\}$ the empirical average of rewards obtained at state $x_{h,i}$ at time t , where $T_{h,i}(t)$ is the number of times that (h, i) has been sampled up to time t .

StoS00 builds an accurate estimate of $f(x_{h,i})$ before (h, i) is expanded. To achieve this, we define an upper confidence bound (or a *b-value*) for each node (h, i) as:

$$b_{h,i}(t) \stackrel{\text{def}}{=} \widehat{\mu}_{h,i}(t) + \sqrt{\frac{\log(Tk/\delta)}{2T_{h,i}(t)}}, \quad (6.1)$$

where δ is the confidence parameter. In the case of $T_{h,i}(t) = 0$, we let $b_{h,i}(t) = \infty$. We refer to $\sqrt{\log(Tk/\delta)/2T_{h,i}(t)}$ as to the *width* of the estimate. Now instead of selecting the promising nodes

according to their values $f(x_{h,i})$, we select them according to their b -values $b_{h,i}$. The analysis of **StoS00** (Valko et al. 2013a) reveals that the simple regret is linked to the depth of the tree after T iterations. This depends on the number of the evaluations per node k before the node is expanded. However, we were only able to deduce the value of k for the partitioning with exponentially decreasing diameters and for the case of near-optimality dimension zero:

Corollary 6.3.1 — Simple regret of **StoS00 for $d = 0$ by Valko et al. 2013a.** For the choice $k = T / \log^3(T)$ and $\delta = 1/\sqrt{T}$, we have:

$$\mathbb{E}[R_T] = O\left(\frac{\log^2 T}{\sqrt{T}}\right).$$

This result shows that, surprisingly, **StoS00** achieves the same rate $\tilde{\mathcal{O}}(T^{-1/2})$, up to a logarithmic factor, as the H00 algorithm run with the best possible metric, although **StoS00** does not require the knowledge of it. While it is not clear how to *adaptively* set k for $d > 0$, in the next section we show how to approach the case $d > 0$ differently, by running several H00 algorithms *in parallel*.

6.4 Parallel optimistic optimization: P00

The **P00** algorithm (Grill et al. 2015) is an algorithm aiming at optimizing functions with unknown smoothness with $d \geq 0$. **P00** uses, as a subroutine, an optimization algorithm that *requires the knowledge* of the function smoothness. We use H00 (Bubeck et al. 2011b) as the base algorithm, but other algorithms, such as HCT (Azar et al. 2014), could be used as well. **P00** runs several H00 instances in parallel, hence the name *parallel optimistic optimization*. The number of base H00 instances and other parameters are adapted to the budget of evaluations and are automatically decided on the fly.

Each instance of H00 requires two real numbers ν and ρ . Running H00 parametrized with (ρ, ν) that are far from the optimal one $(\nu_*, \rho_*)^2$ would cause H00 to underperform. Surprisingly, our analysis of this *suboptimality gap* reveals that it does not decrease too fast as we stray away from (ν_*, ρ_*) . This motivates the following observation. If we *simultaneously* run a slew of H00s with different (ν, ρ) s, one of them is going to perform decently well.

In fact, we show that to achieve a good performance, we only require $\ln T$ H00 instances, where T is the current number of function evaluations. Notice, that we do not require to know the total number of rounds in advance which hints that we can hope for a *naturally anytime* algorithm.

The strategy of **P00** is quite simple: It consists of running N instances of H00 in parallel, that are all launched with different (ν, ρ) s. At the end of the whole process, **P00** selects the instance s^* which performed the best and returns one of the points selected by this instance, chosen uniformly at random. Note that just using a doubling trick in H00 with increasing values of ρ and ν is not enough to guarantee a good performance. Indeed, it is important to keep track of all H00 instances. Otherwise, the regret rate would suffer way too much from using the value of ρ that is too far from the optimal one.

Since **P00** is anytime, the number of instances $N(T)$ is time-dependent and does not need to be known in advance. In fact, $N(T)$ is increased alongside the execution of the algorithm. More precisely, we want to ensure that

$$N(T) \geq \frac{1}{2} D_{\max} \ln(T / \ln T), \quad \text{where} \quad D_{\max} \stackrel{\text{def}}{=} (\ln K) / \ln(1 / \rho_{\max}).$$

To keep the set of different (ν, ρ) s well distributed, the number of H00s is not increased one by one but instead is doubled when needed. Moreover, we also require that H00s run in parallel, perform

²the parameters (ν, ρ) satisfying the assumption from Section 6.1 for which $d(\nu, \rho)$ is the smallest

the same number of function evaluations. Consequently, when we start running new instances, we first ensure to make these instances on par with already existing ones in terms of the number of evaluations.

Finally, as our analysis reveals, a good choice of parameters (ρ_i) is not a uniform grid on $[0, 1]$. Instead, as suggested by our analysis, we require that $1/\ln(1/\rho_i)$ is a uniform grid on $[0, 1/(\ln 1/\rho_{\max})]$. As a consequence, we add H00 instances in batches such that $\rho_i = \rho_{\max}^{N/i}$.

P00 does not require the knowledge of a (v, ρ) verifying the assumption from Section 6.1 and³ yet we prove that it achieves a performance close⁴ to the one obtained by H00 using the best parameters (v_*, ρ_*) . This result solves the open question from the previous section, whether the stochastic optimization of f with unknown parameters (v, ρ) when $d > 0$ for the standard partitioning is possible.

Theorem 6.4.1 — Simple regret of P00 by Grill et al. 2015. Let R_T be the simple regret of P00 at step T . For any (v, ρ) verifying the assumption from Section 6.1 such that $v \leq v_{\max}$ and $\rho \leq \rho_{\max}$ there exists κ such that for all T

$$\mathbb{E}[R_T] \leq \kappa \cdot ((\ln^2 T) / T)^{1/(d(v, \rho)+2)}$$

Moreover, $\kappa = \alpha \cdot D_{\max}(v_{\max}/v_*)^{D_{\max}}$, where α is a constant independent of ρ_{\max} and v_{\max} .

The P00's performance should be compared to the simple regret of H00 run with the best parameters v_* and ρ_* (Theorem 6.2.1). Thus P00's performance is only a factor of $\mathcal{O}((\ln T)^{1/(d(v_*, \rho_*)+2)})$ away from the optimally fitted H00. Furthermore, the regret bound for P00 is slightly better than the regret bound for StoS00 (Corollary 6.3.1) in the case when $d(v, \rho) = 0$ for the same partitioning, i.e., $\mathbb{E}[R_T] = \mathcal{O}(\ln T / \sqrt{T})$. This way P00 generalizes the bound of H00 for any value of $d \geq 0$.

Note that we only give a simple regret bound for P00 whereas H00 ensures a bound on both the cumulative and simple regret.⁵ Notice that since P00 runs several H00s with nonoptimal values of the (v, ρ) parameters, this algorithm explores much more than the optimally fitted H00, which dramatically impacts the cumulative regret. As a consequence, our result applies to the simple regret only.

6.5 Applicability and perspectives of bandit function optimization

In this section, we comment on practical issues when using the methods from this chapter in practice.

Scaling with dimension

The approaches discussed in this chapter strive to optimize the function with *minimal assumptions*. In particular, we showed that it is enough to assume only a local smoothness property around one of the optima and we are able to provide simple regret guarantees. This generality has a *cost*, in particular for scaling with the ambient dimension D . This scaling is exponential and *in general unavoidable*: intuitively, if we split D dimensional hypercube (of the domain of f) along each dimension, the optimum can be in general in any of the sub-hyperrectangles and we need to search each of them. This means that these methods are practically relevant only for a small D .

Hyperparameter optimization

However, these methods can prove very useful for very difficult functions, for which we know very little (*black-box* setting). One good example is hyperparameter optimization, where the *number*

³note that several possible values of those parameters are possible for the same function

⁴up to a logarithmic term $\sqrt{\ln T}$ in the simple regret

⁵in fact, the bound on the simple regret is a direct consequence of the cumulative regret bound (Bubeck et al. 2011a)

of parameters is small and the functions are complex. As an example, `StoS00` was already used in a *Kaggle* (2013) competition *Cause-effect pairs*⁶ in March 2013. The team using `StoS00` (Samothrakis et al. 2013) arrived 3rd (out of 266 teams) and received a prize. Another application is in parameter optimization of simulators which are very costly to run, when the (provable) sample complexity is important.

Extremely difficult functions

Since `H00`, `S00`, `StoS00`, or `P00` require almost no assumptions on f , they can compete with methods that also avoid various smoothness assumptions or existence of derivatives. The most common choices for extremely difficult functions are various *genetic and evolutionary algorithms*. To test the bandit approach to function optimization with them, we participated at their annual *CEC'2014 competition* on single-objective real-parameter numerical optimization test suite (Preux et al. 2014), which showed that on some very difficult functions, `S00` can be competitive while providing performance guarantees.

Extension to other settings

The methods used in this chapter that optimize functions with unknown smoothness could be also used in other settings where we can expect smooth rewards to be present, but where we are unable to quantify this smoothness. One instance is Monte-Carlo planning in MDPs (Grill et al. 2016; Szörényi et al. 2014), where the discount factor naturally induces smoothness among the rewards in distant rounds.

⁶<https://www.kaggle.com/c/cause-effect-pairs>

7. Infinitely many armed bandits

In this chapter, we consider an extension of multi-arm setting to infinitely many actions, where *no topology* (or metric) between the arms is known, the *infinitely many armed bandits* (Berry et al. 1997; Bonald and Proutière 2013; Wang et al. 2008). Inevitably, the sheer amount of possible actions makes it impossible to try each of them even once. Such a setting is practically relevant for cases where one faces a finite, but an extremely large number of actions. This setting was first formalized by Berry et al. (1997) as follows. At each time t , the learner can either sample an arm (a distribution) that has been already observed in the past, or sample a new arm, whose mean μ is sampled from the *mean reservoir distribution* \mathcal{L} .

An example where efficient strategies for minimizing the simple regret of an infinitely many armed bandits are relevant is the search of a good *biomarker* in biology, a single *feature* that performs best on average (Hauskrecht et al. 2006). There can be too many possibilities that we cannot afford to even try each of them in a reasonable time. Our setting is then relevant for this special case of *single feature selection*. In this chapter, we provide the results for the simple regret of an infinitely many armed bandits, a problem that was not considered before. This setting has recently found an application in *hyperparameter optimization* (Lisha Li et al. 2016).

The additional challenges of the infinitely many armed bandits with respect to the multi-armed bandits come from two sources. First, we need to find a good arm among the sampled ones. Second, we need to sample (at least once) enough arms in order to have (at least once) a reasonably good one. These two difficulties ask for a tradeoff which we call the *arm selection tradeoff*. It is different from the known *exploration/exploitation tradeoff* and more linked to model selection principles: On one hand, we want to sample only from a small subsample of arms so that we can decide, with enough accuracy, which one is the best one among them. On the other hand, we want to sample as many arms as possible in order to have a higher chance to sample a good arm at least once. This tradeoff makes the problem of infinitely many armed bandits significantly different from the classic bandit problem.

Berry et al. (1997) provide asymptotic, minimax-optimal (up to a $\log T$ factor) bounds for the *average cumulative regret*, defined as the difference between T times the highest possible value $\bar{\mu}^*$ of the mean reservoir distribution and the mean of the sum of all samples that the learner collects.

A follow-up on this result was the work of Wang et al. (2008), providing algorithms with finite-time regret bounds and the work of Bonald and Proutière (2013), giving an algorithm that is optimal with exact constants in a strictly more specific setting. In all of this prior work, the authors show that it is the *shape* of the arm reservoir distribution what characterizes the *minimax-optimal rate* of the average cumulative regret. Specifically, Berry et al. (1997) and Wang et al. (2008) assume that the mean reservoir distribution is such that, for a small $\varepsilon > 0$, locally around the best arm $\bar{\mu}^*$, we have that

$$\mathbb{P}_{\mu \sim \mathcal{L}}(\bar{\mu}^* - \mu \geq \varepsilon) \approx \varepsilon^\beta, \quad (7.1)$$

that is, they assume that the mean reservoir distribution is β -regularly varying in $\bar{\mu}^*$. When this assumption is satisfied with a known β , their algorithms achieve an expected cumulative regret of order

$$\mathbb{E}[R_T] = \mathcal{O}\left(\max\left(T^{\frac{\beta}{\beta+1}} \text{polylog } T, \sqrt{T} \text{polylog } T\right)\right). \quad (7.2)$$

The limiting factor in the general setting is a $1/\sqrt{T}$ rate for estimating the mean of any of the arms with T samples. This gives the rate (7.2) of \sqrt{T} . It can be refined if the distributions of the arms, that are sampled from the mean reservoir distribution, are Bernoulli of mean μ and $\bar{\mu}^* = 1$ or in the same spirit, if the distributions of the arms are defined on $[0, 1]$ and $\bar{\mu}^* = 1$ as

$$\mathbb{E}[R_T] = \mathcal{O}\left(T^{\frac{\beta}{\beta+1}} \text{polylog } T\right). \quad (7.3)$$

Bonald and Proutière (2013) refine the result (7.3) even more by removing the $\text{polylog } T$ factor and proving upper and lower bounds that *exactly match*, even in terms of constants, for a specific subcase of a uniform mean reservoir distribution. Notice that the rate (7.3) is faster than the more general rate (7.2). This comes from the fact that they assume that the variances of the arms decay with their quality, making finding a good arm easier. For both rates (7.2 and 7.3), β is the *key parameter* for solving the arm selection tradeoff: with smaller β it is more likely that the mean reservoir distribution outputs a high value, and therefore, we need fewer arms for the optimal arm selection tradeoff.

Previous algorithms for this setting were designed for minimizing the cumulative regret of the learner which optimizes the cumulative sum of the rewards. In this chapter, we consider the problem of minimizing the *simple regret*.

7.1 Learning setting

Let $\widetilde{\mathcal{L}}$ be a distribution of distributions. We call $\widetilde{\mathcal{L}}$ the *arm reservoir distribution*, i.e., the distribution of arms. Let \mathcal{L} be the distribution of the means of the distributions output by $\widetilde{\mathcal{L}}$, i.e., the *mean reservoir distribution*. Let \mathbb{A}_t denote the changing set of K_t arms at time t .

At each time $t + 1$, the learner can either choose an arm k_{t+1} among the set of the K_t arms $\mathbb{A}_t = \{v_1, \dots, v_{K_t}\}$ that it has already observed (in this case, $K_{t+1} = K_t$ and $\mathbb{A}_{t+1} = \mathbb{A}_t$), or choose to get a sample of a new arm that is generated according to $\widetilde{\mathcal{L}}$ (in this case, $K_{t+1} = K_t + 1$ and $\mathbb{A}_{t+1} = \mathbb{A}_t \cup \{v_{K_{t+1}}\}$ where $v_{K_{t+1}} \sim \widetilde{\mathcal{L}}$). Let μ_i be the mean of arm i , i.e., the mean of distribution v_i for $i \leq K_t$. We assume that μ_i always exists.

In this setting, the learner observes a sample at each time. At the end of the horizon, which happens at a given time T , the learner has to output an arm $\hat{k} \leq K_T$, and its performance is assessed by the simple regret

$$r_T = \bar{\mu}^* - \mu_{\hat{k}},$$

where $\bar{\mu}^* = \arg \inf_m (\mathbb{P}_{\mu \sim \mathcal{L}}(\mu \leq m) = 1)$ is the right end point of the domain.

Assumption on the samples

The domain of the arm reservoir distribution $\widetilde{\mathcal{L}}$ are distributions of arm samples. We assume that these distributions ν are bounded.

Assumption 1 — Bounded distributions in the domain of $\widetilde{\mathcal{L}}$. Let ν be a distribution in the domain of $\widetilde{\mathcal{L}}$. Then ν is a bounded distribution. Specifically, there exists a universal constant $C > 0$ such that the domain of ν is contained in $[-C, C]$.

This implies that the expectations of all distributions generated by $\widetilde{\mathcal{L}}$ exist, are finite, and bounded by C . In particular, this implies that

$$\bar{\mu}^* = \arg \inf_m (\mathbb{P}_{\mu \sim \mathcal{L}}(\mu \leq m) = 1) < +\infty,$$

which implies that the regret is well defined and that the domain of \mathcal{L} is bounded by $2C$. Note that all the results that we prove hold also for sub-Gaussian distributions ν and bounded \mathcal{L} . Furthermore, it would be possible to relax the sub-Gaussianity using different estimators recently developed for heavy-tailed distributions (Catoni 2012).

Assumption on the arm reservoir distribution

We now assume that the mean reservoir distribution \mathcal{L} has a certain regularity in its right end point, which is a standard assumption for infinitely many armed bandits. Note that this implies that the distribution of the means of the arms is in the domain of attraction of a Weibull distribution, and that it is related to assuming that the distribution is β regularly varying in its end point $\bar{\mu}^*$.

Assumption 2 — β regularity in $\bar{\mu}^*$. Let $\beta > 0$. There exist $\tilde{E}, \tilde{E}' > 0$, and $0 < \tilde{B} < 1$ such that for any $0 \leq \varepsilon \leq \tilde{B}$,

$$\tilde{E}' \varepsilon^\beta \geq \mathbb{P}_{\mu \sim \mathcal{L}}(\mu > \bar{\mu}^* - \varepsilon) \geq \tilde{E} \varepsilon^\beta.$$

This assumption is the same as the classic one (7.1). Standard bounded distributions satisfy Assumption 2 for a specific β , e.g., all the β distributions, in particular the uniform distribution.

We first present the information-theoretic lower bounds for the infinitely many armed bandits with simple regret as the objective. We then present our algorithm and its analysis proving the upper bounds that match the lower bounds — in some cases, depending on β , up to a polylog T factor. This makes our algorithm (almost) *minimax* optimal. Finally, we provide three important extensions.

7.2 Lower bounds

Theorem 7.2.1 exhibits the *information theoretic complexity* of our problem. Comparing these results with the rates for the cumulative regret problem (7.2) from the prior work, one can notice that there are *two regimes* for the cumulative regret results. One regime is characterized by a rate of \sqrt{T} for $\beta \leq 1$, and the other characterized by a $T^{\beta/(1+\beta)}$ rate for $\beta \geq 1$. Both of these regimes are related to the arm selection tradeoff. The first regime corresponds to *easy* problems where the mean reservoir distribution puts a high mass close to $\bar{\mu}^*$, which favors sampling a good arm with high mean from the reservoir. In this regime, the \sqrt{T} rate comes from the parametric $1/\sqrt{T}$ rate for estimating the mean of any arm with T samples. The second regime corresponds to *more difficult* problems where the reservoir is unlikely to output a distribution with a mean close to $\bar{\mu}^*$ and where one has to sample many arms from the reservoir. In this case, the \sqrt{T} rate is not reachable anymore because there are too many arms to choose from subsamples of arms containing good arms. The same dynamics exists also for the *simple regret*, where there are again two regimes, one characterized by a $T^{-1/2}$ rate for $\beta \leq 2$, and the other characterized by a $T^{-1/\beta}$ rate for $\beta \geq 2$.

Provided that these bounds are tight (which is the case, up to a polylog T , Section 7.3), one can see that there is an interesting difference between the cumulative regret problem and the simple regret one. Indeed, the change of regime is here for $\beta = 2$ and not for $\beta = 1$, i.e., the parametric rate of $T^{-1/2}$ is valid for larger values of β for the simple regret. This comes from the fact that for the simple regret objective, there is no exploitation phase and *everything is about exploring*. Therefore, an optimal strategy can spend more time exploring the set of arms and reach the parametric rate also in situations where the cumulative regret does not correspond to the parametric rate.

Theorem 7.2.1 — Simple regret lower bounds for infinitely many arms bandits by Carpenter and Valko 2015. Let us write \mathcal{S}_β for the set of distributions of arms distributions $\tilde{\mathcal{L}}$ that satisfy Assumptions 1 and 2 for the parameters $\beta, \tilde{E}, \tilde{E}', C$. Assume that T is larger than a constant that depends on $\beta, \tilde{E}, \tilde{E}', \tilde{B}, C$. Depending on the value of β , we have the following results, for any algorithm \mathcal{A} , where ν is a small enough constant.

- Case $\beta < 2$: With probability larger than $1/3$,

$$\inf_{\mathcal{A}} \sup_{\tilde{\mathcal{L}} \in \mathcal{S}_\beta} r_T \geq \nu T^{-1/2}.$$

- Case $\beta \geq 2$: With probability larger than $1/3$,

$$\inf_{\mathcal{A}} \sup_{\tilde{\mathcal{L}} \in \mathcal{S}_\beta} r_T \geq \nu T^{-1/\beta}.$$

7.3 SiRI and its upper bounds

In this section, we present our algorithm, the Simple Regret for Infinitely many arms (SiRI).

The SiRI algorithm

Let $b = \min(\beta, 2)$, and let

$$\bar{T}_\beta = \lceil A(T)T^{b/2} \rceil,$$

where

$$A(T) = \begin{cases} A, & \text{if } \beta < 2 \\ A/\log(T)^2, & \text{if } \beta = 2 \\ A/\log(T), & \text{if } \beta > 2 \end{cases}$$

where A is a small constant whose precise value will depend on our analysis. Let \log_2 be the logarithm in base 2. Let us define

$$\bar{i}_\beta = \lfloor \log_2(\bar{T}_\beta) \rfloor.$$

Let $T_{k,t}$ be the number of pulls of arm $k \leq K_t$, and $X_{k,u}$ for the u -th sample of v_k . The empirical mean of the samples of arm k is defined as

$$\hat{\mu}_{k,t} = \frac{1}{T_{k,t}} \sum_{u=1}^{T_{k,t}} X_{k,u}.$$

With this notation, we provide SiRI in Algorithm 4.

Algorithm 4 SiRI *Simple Regret for Infinitely many armed bandits***Parameters:** β, C, δ **Initial pull of arms from the reservoir:**Choose \bar{T}_β arms from the reservoir $\tilde{\mathcal{L}}$.Pull each of \bar{T}_β arms once. $t \leftarrow \bar{T}_\beta$ **Choice between these arms:****while** $t \leq T$ **do**For any $k \leq \bar{T}_\beta$:

$$B_{k,t} \leftarrow \hat{\mu}_{k,t} + 2\sqrt{\frac{C}{T_{k,t}} \log(2^{2\bar{T}_\beta/b}/(T_{k,t}\delta))} + \frac{2C}{T_{k,t}} \log(2^{2\bar{T}_\beta/b}/(T_{k,t}\delta)) \quad (7.4)$$

Pull $T_{k,t}$ times the arm k_t that maximizes $B_{k,t}$ and receive $T_{k,t}$ samples from it. $t \leftarrow t + T_{k,t}$ **end while****Output:** Return the most pulled arm \hat{k} .**Discussion**

SiRI is a UCB-based algorithm, where the leading confidence term is of order

$$\sqrt{\frac{\log(T/(\delta T_{k,t}))}{T_{k,t}}}.$$

Similar to the MOSS algorithm (Audibert and Bubeck 2009), we divide the $\log(\cdot)$ term by $T_{k,t}$, in order to avoid additional logarithmic factors in the bound. But a simpler algorithm with a confidence term as in a classic UCB algorithm for cumulative regret,

$$\sqrt{\frac{\log(T/\delta)}{T_{k,t}}},$$

would provide almost optimal regret, up to a $\log T$, i.e., with a slightly worse regret than what we get. It is quite interesting that with such a confidence term, SiRI is optimal for minimizing the *simple* regret for infinitely many armed bandits, since MOSS, as well as the classic UCB algorithm, targets the cumulative regret. The main difference between our strategy and the cumulative strategies (Berry et al. 1997; Bonald and Proutière 2013; Wang et al. 2008) is in the number of arms sampled from the arm reservoir: For the simple regret, we need to sample more arms. Although the algorithms are related, their analyses are quite different: Our proof (Carpentier and Valko 2015) is *event-based* whereas the proof for the cumulative regret targets *directly the expectations*.

It is also interesting to compare SiRI with existing algorithms targeting the simple regret for finitely many arms, as the ones by Audibert et al. (2010). SiRI can be related to their UCB-E with a specific confidence term and a specific choice of the number of arms selected. Consequently, the two algorithms are related but the regret bounds obtained for UCB-E are not informative when there are infinitely many arms. Indeed, the theoretical performance of UCB-E is decreasing with the sum of the inverse of the gaps squared, which is infinite when there are infinitely many arms. In order to obtain a useful bound, in this case, we need to consider a more refined analysis which is the one that leads to Theorem 7.3.1.

Main result

We now state the main result which characterizes SiRI's simple regret according to β .

Theorem 7.3.1 — Upper bounds of SiRI by Carpentier and Valko 2015. Let $\delta > 0$. Assume all Assumptions 1 and 2 of the model and that T is larger than a large constant that depends on $\beta, \tilde{E}, \tilde{E}', \tilde{B}, C$. Depending on the value of β , we have the following results, where E is a large enough constant.

- Case $\beta < 2$: With probability larger than $1 - \delta$,

$$r_T \leq ET^{-1/2} \log(1/\delta) (\log(\log(1/\delta)))^{96} \sim T^{-1/2}.$$

- Case $\beta > 2$: With probability larger than $1 - \delta$,

$$r_T \leq E(T \log(T))^{-1/\beta} (\log(\log(\log(T)/\delta)))^{96} \log(\log(T)/\delta) \sim (T \log T)^{-1/\beta} \text{polyloglog } T.$$

- Case $\beta = 2$: With probability larger than $1 - \delta$,

$$r_T \leq E \log(T) T^{-1/2} (\log(\log(\log(T)/\delta)))^{96} \log(\log(T)/\delta) \sim T^{-1/2} \log T \text{polyloglog } T.$$

Upper bounds discussion

The bound we obtain is minimax optimal for $\beta < 2$ *without additional $\log T$ factors*. We emphasize it since the previous results on infinitely many armed bandits give results which are optimal up to a $\text{polylog } T$ factor for the cumulative regret, except the one by Bonald and Proutière (2013) which considers a very specific and fully parametric setting. For $\beta \geq 2$, our result is optimal up to a $\text{polylog } T$ factor. We conjecture that the lower bound of Theorem 7.2.1 for $\beta \geq 2$ can be improved to $(\log(T)/T)^{1/\beta}$ and that SiRI is actually optimal up to a $\text{polyloglog}(T)$ factor for $\beta > 2$.

7.4 Extensions of SiRI

We provided also three important extensions (Carpentier and Valko 2015). The first extension concerns the case where the distributions of the arms are defined on $[0, 1]$ and where $\bar{\mu}^* = 1$. In this case, replacing the Hoeffding bound in the confidence term of our algorithm by a Bernstein bound, bounds the simple regret as

$$r_T = \mathcal{O} \left(\max \left(\frac{1}{T} \text{polylog } T, (T \log T)^{-\frac{1}{\beta}} \text{polyloglog } T \right) \right).$$

The second extension treats *unknown β* . We prove (Carpentier and Valko 2015) that it is possible to estimate β with enough precision, so that its knowledge is not necessary for implementing the algorithm. This can also be applied to the prior work (Berry et al. 1997; Wang et al. 2008) where β is also necessary for implementation and optimal bounds. Finally, in the third extension, we make the algorithm anytime using known tools (Carpentier and Valko 2015).



References

- Abbasi-Yadkori, Yasin, Dávid Pál, and Csaba Szepesvári (2011). “Improved algorithms for linear stochastic bandits”. In: *Neural Information Processing Systems* (cited on pages 9, 18).
- Agrawal, Shipra and Navin Goyal (2013). “Thompson sampling for contextual bandits with linear payoffs”. In: *International Conference on Machine Learning* (cited on pages 9, 16, 18).
- Alaoui, Ahmed El and Michael W. Mahoney (2015). “Fast randomized kernel methods with statistical guarantees”. In: *Neural Information Processing Systems* (cited on page 53).
- Alon, Noga, Nicolò Cesa-Bianchi, Ofer Dekel, and Tomer Koren (2015). “Online learning with feedback graphs: Beyond bandits”. In: *Conference on Learning Theory* (cited on pages 24, 28, 34, 39).
- Alon, Noga, Nicolò Cesa-Bianchi, Claudio Gentile, Shie Mannor, Yishay Mansour, and Ohad Shamir (2014). *Nonstochastic multi-armed bandits with graph-structured feedback*. Technical report. arXiv: 1409.8428 (cited on page 34).
- Alon, Noga, Nicolò Cesa-Bianchi, Claudio Gentile, and Yishay Mansour (2013). “From bandits to experts: A tale of domination and independence”. In: *Neural Information Processing Systems* (cited on pages 8, 24, 26–28, 33–35, 39).
- Ashkan, Azin, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen (2014). “Diversified utility maximization for recommendations”. In: *Conference on Recommender Systems* (cited on page 57).
- (2015). “Optimal greedy diversity for recommendation”. In: *International Joint Conferences on Artificial Intelligence* (cited on page 57).
- Audibert, Jean-Yves and Sébastien Bubeck (2009). “Minimax policies for adversarial and stochastic bandits”. In: *Conference on Learning Theory* (cited on page 75).
- Audibert, Jean-Yves, Sébastien Bubeck, and Gábor Lugosi (2014). “Regret in online combinatorial optimization”. In: *Mathematics of Operations Research* 39, pages 31–45 (cited on pages 28, 59).
- Audibert, Jean-Yves, Sébastien Bubeck, and Rémi Munos (2010). “Best arm identification in multi-armed bandits”. In: *Conference on Learning Theory* (cited on page 75).

- Auer, Peter (2002). “Using confidence bounds for exploitation-exploration trade-offs”. In: *Journal of Machine Learning Research* 3, pages 397–422 (cited on pages 9, 16, 19, 49, 51).
- Auer, Peter, Nicolò Cesa-Bianchi, and Paul Fischer (2002a). “Finite-time analysis of the multiarmed bandit problem”. In: *Machine Learning* 47.2-3, pages 235–256 (cited on pages 7, 59, 63).
- Auer, Peter, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire (2002b). “The non-stochastic multi-armed bandit problem”. In: *Journal on Computing* 32.1, pages 48–77 (cited on pages 13, 27, 36, 60, 61).
- Auer, Peter and Ronald Ortner (2010). “UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem”. In: *Periodica Mathematica Hungarica* (cited on page 18).
- Azar, Mohammad Gheshlaghi, Alessandro Lazaric, and Emma Brunskill (2014). “Online Stochastic Optimization under Correlated Bandit Feedback”. In: *International Conference on Machine Learning* (cited on pages 63–67).
- Azuma, Kazuoki (1967). “Weighted sums of certain dependent random variables”. In: *Tohoku Mathematical Journal* 19.3, pages 357–367 (cited on page 18).
- Bala, Venkatesh and Sanjeev Goyal (1998). “Learning from neighbours”. In: *Review of Economic Studies* 65.3, pages 595–621 (cited on page 34).
- (2001). “Conformism and diversity under social learning”. In: *Economic Theory* 17, pages 101–120 (cited on page 34).
- Bao, Yixin, Xiaoke Wang, Zhi Wang, Chuan Wu, and Francis C. M. Lau (2016). “Online influence maximization in non-stationary social networks”. In: *International Symposium on Quality of Service* (cited on page 44).
- Barabási, Albert-László and Réka Albert (1999). “Emergence of scaling in random networks”. In: *Science* 286, page 11 (cited on pages 37, 41).
- Bartók, Gábor, Dean P. Foster, Dávid Pál, Alexander Rakhlin, and Csaba Szepesvári (2014). “Partial monitoring-classification, regret bounds, and algorithms”. In: *Mathematics of Operations Research* 39.4, pages 967–997 (cited on page 34).
- Bartók, Gábor, Dávid Pál, and Csaba Szepesvári (2011). “Minimax regret of finite partial-monitoring games in stochastic environments”. In: *Conference on Learning Theory* (cited on page 34).
- Belkin, Mikhail, Irina Matveeva, and Partha Niyogi (2004). “Regularization and semi-supervised learning on large graphs”. In: *Conference on Learning Theory* (cited on page 15).
- Belkin, Mikhail, Partha Niyogi, and Vikas Sindhwani (2006). “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples”. In: *Journal of Machine Learning Research* 7, pages 2399–2434 (cited on page 15).
- Berry, Donald A., Robert W. Chen, Alan Zame, David C. Heath, and Larry A. Shepp (1997). “Bandit problems with infinitely many arms”. In: *Annals of Statistics* 25, pages 2103–2116 (cited on pages 9, 47, 71, 72, 75, 76).
- Bertsimas, Dimitris and John Tsitsiklis (1997). *Introduction to linear optimization*. Athena Scientific (cited on page 57).
- Billsus, Daniel, Michael J. Pazzani, and James Chen (2000). “A learning agent for wireless news access”. In: *International Conference on Intelligent User Interfaces* (cited on page 15).
- Bnaya, Zahy, Rami Puzis, Roni Stern, and Ariel Felner (2013). “Social network search as a volatile multi-armed bandit problem”. In: *Human Journal* 2.2, pages 84–98 (cited on page 44).
- Bonald, Thomas and Alexandre Proutière (2013). “Two-target algorithms for infinite-armed bandits with Bernoulli rewards”. In: *Neural Information Processing Systems* (cited on pages 71, 72, 75, 76).
- Bubeck, Sébastien, Rémi Munos, and Gilles Stoltz (2011a). “Pure exploration in finitely-armed and continuous-armed bandits”. In: *Theoretical Computer Science* 412.19, pages 1832–1852 (cited on pages 66, 68).

- Bubeck, Sébastien, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári (2011b). “ \mathcal{X} -armed bandits”. In: *Journal of Machine Learning Research* 12, pages 1587–1627 (cited on pages 63–67).
- Buccapatnam, Swapna, Atila Eryilmaz, and Ness B. Shroff (2014). “Stochastic bandits with side observations on networks”. In: *International Conference on Measurement and Modeling of Computer Systems* (cited on pages 33, 39).
- Bull, Adam D. (2015). “Adaptive-treed bandits”. In: *Bernoulli* 21.4, pages 2289–2307 (cited on pages 63, 64).
- Burnetas, Apostolos N. and Michaël N. Katehakis (1996). “Optimal adaptive policies for sequential allocation problems”. In: *Advances in Applied Mathematics* 17(2), pages 122–142 (cited on page 7).
- Calandriello, Daniele, Alessandro Lazaric, and Michal Valko (2016). “Analysis of Nyström method with sequential ridge leverage scores”. In: *Uncertainty in Artificial Intelligence* (cited on page 53).
- Caron, Stéphane, Branislav Kveton, Marc Lelarge, and Smriti Bhagat (2012). “Leveraging side observations in stochastic bandits.” In: *Uncertainty in Artificial Intelligence* (cited on pages 33, 39).
- Carpentier, Alexandra and Michal Valko (2015). “Simple regret for infinitely many armed bandits”. In: *International Conference on Machine Learning* (cited on pages 9, 74–76).
- (2016). “Revealing graph bandits for maximizing local influence”. In: *International Conference on Artificial Intelligence and Statistics* (cited on pages 8, 42).
- Catoni, Olivier (2012). “Challenging the empirical mean and empirical variance: A deviation study”. In: *Annales de l’Institut Henri Poincaré, Probabilités et Statistiques*. Volume 48. 4, pages 1148–1185 (cited on page 73).
- Cesa-Bianchi, Nicolò, Claudio Gentile, Yishay Mansour, and Alberto Minora (2016). “Delay and cooperation in nonstochastic bandits”. In: *Conference on Learning Theory* (cited on page 34).
- Cesa-Bianchi, Nicolò, Claudio Gentile, and Giovanni Zappella (2013). “A gang of bandits”. In: *Neural Information Processing Systems* (cited on pages 20, 22, 39).
- Cesa-Bianchi, Nicolò and Gábor Lugosi (2006). *Prediction, learning, and games*. Cambridge University Press (cited on page 13).
- (2012). “Combinatorial bandits”. In: *Journal of Computer and System Sciences*. Volume 78. 5, pages 1404–1422 (cited on pages 28, 29).
- Chau, Duen Horng, Aniket Kittur, Jason I. Hong, and Christos Faloutsos (2011). “Apolo: Making sense of large network data by combining rich user interaction and machine learning”. In: *Conference on Human Factors in Computing Systems* (cited on page 15).
- Chen, Wei, Chi Wang, and Yajun Wang (2010). “Scalable influence maximization for prevalent viral marketing in large-scale social networks”. In: *Knowledge Discovery and Data Mining* (cited on page 44).
- Chen, Wei, Yajun Wang, and Yang Yuan (2016). “Combinatorial multi-armed bandit and its extension to probabilistically triggered arms”. In: *Journal of Machine Learning Research* 17 (cited on page 43).
- Chu, Lei, Lihong Li, Lev Reyzin, and Robert E Schapire (2011). “Contextual bandits with linear payoff functions”. In: *International Conference on Artificial Intelligence and Statistics* (cited on pages 19, 51–53).
- Cohen, Alon, Tamir Hazan, and Tomer Koren (2016). “Online learning with feedback graphs without the graphs”. In: *International Conference on Machine Learning* (cited on page 35).
- Combes, Richard and Alexandre Proutière (2014). “Unimodal bandits: Regret lower bounds and optimal algorithms”. In: *International Conference on Machine Learning* (cited on page 21).
- Contal, Emile and Nicolas Vayatis (2016). *Stochastic process bandits: Upper confidence bounds algorithms via generic chaining*. Technical report. arXiv: 1602.04976 (cited on page 53).

- Coquelin, Pierre-Arnaud and Rémi Munos (2007). “Bandit algorithms for tree search”. In: *Uncertainty in Artificial Intelligence* (cited on page 63).
- Coulom, Rémi (2007). “Efficient selectivity and backup operators in Monte-Carlo tree search”. In: *Computers and games* 4630, pages 72–83 (cited on page 63).
- Desautels, Thomas, Andreas Krause, and Joel Burdick (2012). “Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization”. In: *International Conference on Machine Learning* (cited on page 19).
- Edmonds, Jack (1970). “Submodular functions, matroids, and certain polyhedra”. In: *Combinatorial Structures and Their Applications*, pages 69–87 (cited on page 56).
- Ellison, Glenn and Drew Fudenberg (1993). “Rules of thumb for social learning”. In: *Journal of Political Economy* 101.4, pages 612–643 (cited on page 34).
- Erdős, Paul and Alfréd Rényi (1959). “On random graphs”. In: *Publicationes Mathematicae* 6, pages 290–297 (cited on page 35).
- Fang, Meng and Dacheng Tao (2014). “Networked bandits with disjoint linear payoffs”. In: *International Conference on Knowledge Discovery and Data Mining* (cited on page 21).
- Farajtabar, Mehrdad, Xiaojing Ye, Sahar Harati, Le Song, and Hongyuan Zha (2016). “Multistage campaigning in social networks”. In: *Neural Information Processing Systems* (cited on page 44).
- Fujishige, Satoru (2005). *Submodular functions and optimization*. Annals of discrete mathematics (cited on page 57).
- Gabillon, Victor, Branislav Kveton, Zheng Wen, Brian Eriksson, and S. Muthukrishnan (2013). “Adaptive submodular maximization in bandit setting”. In: *Neural Information Processing Systems* (cited on page 61).
- (2014). “Large-scale optimistic adaptive submodularity”. In: *AAAI Conference on Artificial Intelligence* (cited on page 61).
- Gai, Yi, Bhaskar Krishnamachari, and Rahul Jain (2012). “Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations”. In: *Transactions on Networking* 20.5, pages 1466–1478 (cited on page 60).
- Gale, Douglas and Shachar Kariv (2003). “Bayesian learning in social networks”. In: *Games and Economic Behavior* 45.2, pages 329–346 (cited on page 34).
- Gelly, Sylvain, Wang Yizao, Rémi Munos, and Olivier Teytaud (2006). *Modification of UCT with patterns in Monte-Carlo Go*. Technical report. Inria. URL: <https://hal.inria.fr/inria-00117266> (cited on page 63).
- Gentile, Claudio, Shuai Li, and Giovanni Zappella (2014). “Online clustering of bandits”. In: *International Conference on Machine Learning* (cited on pages 21, 39).
- Ghosh, Shaona and Adam Prügél-Bennett (2015). “Ising bandits with side Information”. In: *European Conference on Machine Learning* (cited on page 34).
- Girvan, Michelle and Mark E J Newman (2002). “Community structure in social and biological networks.” In: *National Academy of Sciences of the United States of America* 99.12, pages 7821–6 (cited on page 37).
- Grill, Jean-Bastien, Michal Valko, and Rémi Munos (2015). “Black-box optimization of noisy functions with unknown smoothness”. In: *Neural Information Processing Systems* (cited on pages 9, 63–65, 67, 68).
- (2016). “Blazing the trails before beating the path: Sample-efficient Monte-Carlo planning”. In: *Neural Information Processing Systems* (cited on page 69).
- Gu, Quanquan and Jiawei Han (2014). “Online spectral learning on a graph with bandit feedback”. In: *International Conference on Data Mining* (cited on pages 20, 39).
- Guillory, Andrew and Jeff Bilmes (2011). “Online submodular set cover, ranking, and repeated active learning”. In: *Neural Information Processing Systems* (cited on page 61).

-
- Guillou, Frédéric, Romaric Gaudel, and Philippe Preux (2015). “Collaborative filtering as a multi-armed bandit”. In: *NIPS Workshop on Machine Learning for eCommerce* (cited on page 22).
- (2016). “Scalable explore-exploit collaborative filtering”. In: *Pacific Asia Conference on Information Systems* (cited on page 22).
- Haasdonk, Bernard and Elżbieta Pełalska (2010). “Classification with kernel Mahalanobis distance classifiers”. In: *Advances in Data Analysis, Data Handling and Business Intelligence*, pages 351–361 (cited on pages 49, 50).
- Hanawal, Manjesh, Venkatesh Saligrama, Michal Valko, and Rémi Munos (2015). “Cheap bandits”. In: *International Conference on Machine Learning* (cited on pages 8, 20).
- Hannan, James (1957). “Approximation to Bayes risk in repeated play”. In: *Contributions to the theory of games* 3, pages 97–139 (cited on page 29).
- Hauskrecht, Milos, Richard Pelikan, Michal Valko, and James Lyons-Weiler (2006). “Feature selection and dimensionality reduction in genomics and proteomics”. In: *Fundamentals of Data Mining in Genomics and Proteomics*. Springer (cited on page 71).
- Jannach, Dietmar, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich (2010). *Recommender systems: An introduction*. Cambridge University Press (cited on page 15).
- Kaggle (2013). URL: <https://www.kaggle.com/> (cited on page 69).
- Kalai, Adam and Santosh Vempala (2005). “Efficient algorithms for online decision problems”. In: *Journal of Computer and System Sciences* 71.3, pages 291–307 (cited on page 29).
- Kawale, Jaya, Hung Hai Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla (2015). “Efficient Thompson sampling for online matrix-factorization recommendation”. In: *Neural Information Processing Systems* (cited on page 22).
- Kempe, David, Jon Kleinberg, and Éva Tardos (2003). “Maximizing the spread of influence through a social network”. In: *Knowledge Discovery and Data Mining*, page 137 (cited on pages 39, 43).
- (2015). “Maximizing the spread of influence through a social network”. In: *Theory of Computing* 11.4, pages 105–147 (cited on pages 8, 39).
- Kleinberg, Robert, Aleksandrs Slivkins, and Eli Upfal (2008). “Multi-armed bandit problems in metric spaces”. In: *Symposium on Theory Of Computing* (cited on pages 63, 64, 66).
- Kocák, Tomáš, Gergely Neu, and Michal Valko (2016a). “Online learning with Erdős-Rényi side-observation graphs”. In: *Uncertainty in Artificial Intelligence* (cited on pages 35, 36).
- (2016b). “Online learning with noisy side observations”. In: *International Conference on Artificial Intelligence and Statistics* (cited on pages 8, 31, 33).
- Kocák, Tomáš, Gergely Neu, Michal Valko, and Rémi Munos (2014a). “Efficient learning by implicit exploration in bandit problems with side observations”. In: *Neural Information Processing Systems* (cited on pages 8, 24, 27, 28, 30, 32, 35, 39).
- Kocák, Tomáš, Michal Valko, Rémi Munos, and Shipra Agrawal (2014b). “Spectral Thompson sampling”. In: *AAAI Conference on Artificial Intelligence* (cited on pages 8, 19).
- Kocsis, Levente and Csaba Szepesvári (2006). “Bandit based Monte-Carlo planning”. In: *European Conference on Machine Learning* (cited on pages 63, 65).
- Kolla, Ravi Kumar, Krishna Jagannathan, and Aditya Gopalan (2016). “Collaborative learning of stochastic bandits over a social network”. In: *Annual Allerton Conference on Communication, Control, and Computing* (cited on page 33).
- Koolen, Wouter M., Manfred K. Warmuth, and Jyrki Kivinen (2010). “Hedging structured concepts”. In: *Conference on Learning Theory* (cited on pages 28, 29).
- Korda, Nathan, Balázs Szörényi, and Shuai Li (2016). “Distributed clustering of linear bandits in peer to peer networks”. In: *International Conference on Machine Learning* (cited on page 21).

- Koutis, Ioannis, Gary L. Miller, and David Tolliver (2011). “Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing”. In: *Computer Vision and Image Understanding* 115.12, pages 1638–1646 (cited on page 19).
- Kveton, Branislav, Zheng Wen, Azin Ashkan, Hoda Eydgahi, and Brian Eriksson (2014). “Matroid bandits: Fast combinatorial optimization with learning”. In: *Uncertainty in Artificial Intelligence* (cited on pages 58–60).
- Kveton, Branislav, Zheng Wen, Azin Ashkan, and Michal Valko (2016). “Learning to act greedily: Polymatroid semi-bandits”. In: *Journal of Machine Learning Research* (cited on pages 9, 47, 60, 61).
- Lei, Siyu, Silviu Maniu, Luyi Mo, Reynold Cheng, and Pierre Senellart (2015). “Online influence maximization”. In: *Knowledge Discovery and Data Mining* (cited on page 43).
- Li, Lihong, Wei Chu, John Langford, and Robert E. Schapire (2010). “A contextual-bandit approach to personalized news article recommendation”. In: *International World Wide Web Conference* (cited on pages 9, 16–18).
- Li, Lisha, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar (2016). *Efficient hyperparameter optimization and infinitely many armed bandits*. Technical report. arXiv: 1603.06560 (cited on page 71).
- Li, Shuai, Alexandros Karatzoglou, and Claudio Gentile (2016). “Collaborative filtering Bandits”. In: *Conference on Research and Development in Information Retrieval* (cited on page 21).
- Ma, Yifei, Tzu-Kuo Huang, and Jeff Schneider (2015). “Active search and bandits on graphs using sigma-optimality”. In: *Uncertainty in Artificial Intelligence* (cited on page 20).
- Mannor, Shie and Ohad Shamir (2011). “From bandits to experts: On the value of side-observations”. In: *Neural Information Processing Systems* (cited on pages 8, 23–26, 32–35, 39).
- Mary, Jérémie, Romaric Gaudel, and Philippe Preux (2015). “Bandits and recommender systems”. In: *First International Workshop on Machine Learning, Optimization, and Big Data* (cited on page 22).
- McPherson, Miller, Lynn Smith-Lovin, and James Cook (2001). “Birds of a feather: Homophily in social networks”. In: *Annual Review of Sociology* 27, pages 415–444 (cited on page 15).
- Megiddo, Nimrod (1974). “Optimal flows in networks with multiple sources and sinks”. In: *Mathematical Programming* 7.1, pages 97–107 (cited on page 55).
- Munos, Rémi (2011). “Optimistic optimization of deterministic functions without the knowledge of its smoothness”. In: *Neural Information Processing Systems* (cited on page 66).
- (2014). “From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning”. In: *Foundations and Trends in Machine Learning* 7(1), pages 1–130 (cited on page 63).
- Narang, Sunil K., Akshay Gadde, and Antonio Ortega (2013). “Signal processing techniques for interpolation in graph structured data”. In: *International Conference on Acoustics, Speech and Signal Processing* (cited on page 20).
- Narasimhan, Harikrishna, David C. Parkes, and Yaron Singer (2015). “Learnability of influence in networks”. In: *Neural Information Processing Systems* (cited on page 44).
- Neu, Gergely (2015). “Explore no more: Improved high-probability regret bounds for non-stochastic bandits”. In: *Neural Information Processing Systems* (cited on page 34).
- Neu, Gergely and Gábor Bartók (2013). “An efficient algorithm for learning with semi-bandit feedback”. In: *Algorithmic Learning Theory* (cited on page 30).
- Papadimitriou, Christos and Kenneth Steiglitz (1998). *Combinatorial Optimization*. Dover Publications (cited on page 61).
- Preux, Philippe, Rémi Munos, and Michal Valko (2014). “Bandits attack function optimization”. In: *Congress on Evolutionary Computation* (cited on pages 9, 64, 69).

-
- Prisadnikov, Nedyalko (2014). *Exploration-exploitation trade-offs via probabilistic matrix factorization*. Technical report. Master Thesis, ETH-Zürich, Department of Computer Science. DOI: [10.3929/ethz-a-010211630](https://doi.org/10.3929/ethz-a-010211630) (cited on page 22).
- Rudi, Alessandro, Raffaello Camoriano, and Lorenzo Rosasco (2015). “Less is more: Nyström computational regularization”. In: *Neural Information Processing Systems* (cited on page 53).
- Samothrakis, Spyridon, Diego Perez, and Simon Lucas (2013). “Training gradient boosting machines using curve-fitting and information-theoretic features for causal direction detection”. In: *NIPS Workshop on Causality* (cited on page 69).
- Schölkopf, Bernhard and Alexander J. Smola (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press (cited on page 53).
- Seldin, Yevgeny, Peter Bartlett, Koby Crammer, and Yasin Abbasi-Yadkori (2014). “Prediction with limited advice and multiarmed bandits with paid observations”. In: *International Conference on Machine Learning* (cited on page 36).
- Shawe-Taylor, John and Nelo Cristianini (2004). *Kernel methods for pattern analysis*. Cambridge University Press (cited on pages 49, 51, 53).
- Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis (2016). “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587, pages 484–489 (cited on page 63).
- Singla, Adish, Eric Horvitz, Pushmeet Kohli, Ryen White, and Andreas Krause (2015). “Information gathering in networks via active exploration”. In: *International Joint Conferences on Artificial Intelligence* (cited on page 44).
- Slivkins, Aleksandrs (2011). “Multi-armed bandits on implicit metric spaces”. In: *Neural Information Processing Systems* (cited on pages 64, 65).
- Srinivas, Niranjan, Andreas Krause, Sham M. Kakade, and Matthias Seeger (2010). “Gaussian process optimization in the bandit setting: No regret and experimental design”. In: *International Conference on Machine Learning* (cited on pages 9, 16, 47, 52, 53).
- Szörényi, Balázs, Gunnar Kedenburg, and Rémi Munos (2014). “Optimistic planning in Markov decision processes using a generative model”. In: *Neural Information Processing Systems* (cited on page 69).
- Thompson, William R. (1933). “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples”. In: *Biometrika* 25, pages 285–294 (cited on page 19).
- Tu, Shi-Tao and Juan-Lan Zhu (2015). “A bandit method using probabilistic matrix factorization in recommendation”. In: *Journal of Shanghai Jiaotong University (Science)* 20.5, pages 535–539 (cited on page 22).
- Valko, Michal, Alexandra Carpentier, and Rémi Munos (2013a). “Stochastic simultaneous optimistic optimization”. In: *International Conference on Machine Learning* (cited on pages 9, 65, 67).
- Valko, Michal, Nathaniel Korda, Rémi Munos, Ilias Flaounas, and Nelo Cristianini (2013b). “Finite-Time Analysis of Kernelised Contextual Bandits”. In: *Uncertainty in Artificial Intelligence* (cited on pages 9, 16, 47, 52).
- Valko, Michal, Rémi Munos, Branislav Kveton, and Tomáš Kocák (2014). “Spectral bandits for smooth graph functions”. In: *International Conference on Machine Learning* (cited on pages 7, 9, 18, 21, 39).
- Vaswani, Sharan and Laks V. S. Lakshmanan (2016). *Adaptive influence maximization in social networks: Why commit when you can adapt?* Technical report. arXiv: [1604.08171](https://arxiv.org/abs/1604.08171) (cited on page 44).

- Vaswani, Sharan, Laks. V. S. Lakshmanan, and Mark Schmidt (2015). “Influence maximization with bandits”. In: *NIPS workshop on Networks in the Social and Information Sciences 2015* (cited on page 44).
- Wang, Yizao, Jean-Yves Audibert, and Rémi Munos (2008). “Algorithms for infinitely many-armed bandits”. In: *Neural Information Processing Systems* (cited on pages 71, 72, 75, 76).
- Watts, Duncan J. and Steven H. Strogatz (1998). “Collective dynamics of small-world networks”. In: *Nature* 393, pages 440–442 (cited on page 37).
- Wen, Zheng, Branislav Kveton, Brian Eriksson, and Sandilya Bhamidipati (2013). “Sequential Bayesian search”. In: *International Conference on Machine Learning* (cited on page 61).
- Wen, Zheng, Branislav Kveton, and Michal Valko (2016). *Influence maximization with semi-bandit feedback*. Technical report. arXiv: 1605.06593 (cited on page 44).
- Whitney, Hassler (1935). “On the abstract properties of linear dependence”. In: *American Journal of Mathematics* 57.3, pages 509–533 (cited on page 57).
- Wu, Yifan, András György, and Csaba Szepesvári (2015). “Online learning with Gaussian payoffs and side observations”. In: *Neural Information Processing Systems* (cited on pages 31, 33, 34).
- Yu, Jia Yuan and Shie Mannor (2011). “Unimodal bandits”. In: *International Conference on Machine Learning* (cited on pages 21, 39).
- Yue, Yisong and Carlos Guestrin (2011). “Linear submodular bandits and their application to diversified retrieval”. In: *Neural Information Processing Systems* (cited on page 61).
- Zhang, Fuzhen (2005). *The Schur complement and its applications*. Volume 4. Springer (cited on page 19).
- Zhu, Xiaojin (2008). *Semi-supervised learning literature survey*. Technical report 1530. University of Wisconsin-Madison (cited on page 15).