Online Submodular Minimization for Combinatorial Structures

Stefanie Jegelka JEGELKA@TUEBINGEN.MPG.DE
Max Planck Institute for Intelligent Systems, Tübingen, Germany

Jeff Bilmes BILMES@U.WASHINGTON.EDU
University of Washington, Seattle, WA 98195, USA

Abstract

Most results for online decision problems with structured concepts, such as trees or cuts, assume linear costs. In many settings, however, nonlinear costs are more realistic. Owing to their non-separability, these lead to much harder optimization problems. Going beyond linearity, we address online approximation algorithms for structured concepts that allow the cost to be submodular, i.e., nonseparable. In particular, we show regret bounds for three Hannan-consistent strategies that capture different settings. Our results also tighten a regret bound for unconstrained online submodular minimization.

1. Introduction

Online decision problems require repeatedly choosing a solution S with knowledge only of the costs for previous decisions. The goal is to be in the long run competitive to the best solution in hindsight. This online setting becomes more challenging if in each round, a combinatorial *structure* must be chosen as the solution, e.g., a path or spanning tree (Kalai & Vempala, 2005; Koolen et al., 2010; Kakade et al., 2009). Almost always, the cost function is assumed to be linear, i.e., separable. Often, this separability is key to tackle the combinatorial explosion of the decision space which otherwise poses difficulties to methods based on maintaining weights. Indeed, non-separability of the cost usually renders the offline problem NP-hard. Then only approximation algorithms are possible, a further complication. Accordingly, and despite the broad interest that online problems have enjoyed in Machine Learning, results for online combinatorial problems with nonlinear costs are quite scarce, and mostly relate to simple constraints or problems that are solvable exactly. But the restriction to separable costs fails to capture several real-world situations. Therefore, we address online algorithms for nonseparable, submodular costs.

In the online combinatorial setting, we have a ground set E of elements, e.g., the edges in a fixed graph. In each of T rounds, we must choose a structure S_t from a family $\mathcal{S} \subset 2^E$ that contains, e.g., all spanning trees, or all matchings. Then the cost function f_t is revealed that determines the loss $f_t(S_t)$. If we knew f_t, we would solve the problem

$$\min f_t(S) \quad \text{subject to } S \in \mathcal{S}. \tag{1}$$

Examples are routing or connectivity problems, where the graph structure does not change over time, but the cost $f : 2^E \to \mathbb{R}_+$ of the edges does. Usually, the cost function is a sum $w(S) = \sum_{e \in S} w(e)$ of nonnegative edge weights $w : E \to \mathbb{R}_+$. With this cost, Problem (1) reduces to a well-studied problem like shortest path or minimum spanning tree.

The sum of weights, however, fails to capture the cost in many real-life situations. As an example, edges in a graph might be operated by different companies, and there is a discount for using the same company on many edges. Similarly, some elements might have shared fixed costs or depend on shared resources. Such discounts are captured by *submodular* set functions. Thus, we allow f to be a nondecreasing submodular set function. A function $f : 2^E \to \mathbb{R}_+$ is submodular if it satisfies *diminishing marginal costs*: for any $A \subseteq B \subseteq E \setminus \{e\}$, it holds that $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$. The function is nondecreasing if for all $A \subseteq B \subseteq E$, it holds that $f(A) \leq f(B)$. A sum of weights satisfies diminishing costs with equality, and is thus called a *modular* function. All separable functions are modular. As opposed to submodular functions, modular functions cannot express dependencies between the costs of two elements. Recently, theoretical computer science has seen a rising interest in combinatorial problems like (1)

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.
Copyright 2011 by the author(s)/owner(s).

with submodular costs (Iwata & Nagano, 2009; Goel et al., 2009; Koufogiannakis & Young, 2009). Such submodular problems arise in a variety of applications, such as the following.

Label Costs. Each element $e \in E$ has a set of labels (features) $\pi(e) \subset \mathcal{L}$, and the cost $f(S)$ of a set $S \subseteq E$ is the cost of the labels of its elements: $f(S) = c(\bigcup_{e \in S} \pi(e))$. Labels are shared among several elements, and the cost of a set of labels L is additive: $c(L) = \sum_{\ell \in L} c(\ell)$ (ref. in (Hassin et al., 2007)). In a network, the labels can correspond to transportation media or edges maintained by the same company, and choosing paths or trees spanning few labels lowers the cost. Another application is reliable connectivity structures in networks: links do not break independently, but share physical resources or other common sources of failure. They belong to "Shared Risk Link Groups" (Yuan et al., 2005), modeled by common labels. Related ideas have surfaced in computer security, where the minimum label cut in an attack graph indicates the lowest-cost prevention of an intrusion (Jha et al., 2002). Finally, the "multiple query optimization problem" can be phrased as a label cost problem. In all these examples, submodular costs allow even the cost of the labels themselves to enjoy discounts.

Minimum power assignment. In wireless ad-hoc networks, we seek a connectivity structure (e.g., a spanning tree) that has minimum power requirement. The power consumption of a node depends on the most expensive edge it is using, $p(v|S) = \max_{e=(v,u) \in S} c(e)$, and the total cost is the sum of the node costs, $f(S) = \sum_{v \in V} p(v|S)$ (Calinescu et al., 2003).

Image segmentation. Graph cuts are a versatile tool in computer vision. Submodular edge costs allow to couple edges and improve segmentations in difficult settings (Jegelka & Bilmes, 2011a).

Stochastic optimization. In discrete mean-risk minimization, we aim to minimize a stochastic cost function over $\mathcal{S}$ while avoiding risks. The resulting optimization problems have cost functions of the form $f(S) = \sum_{e \in S} \mu_e + \Omega \sqrt{\sum_{e \in S} \sigma_e^2}$ – a submodular function. Instead of the root, other concave functions can arise that yield submodular set functions (Atamtürk & Narayanan, 2008).

1.1. Online setting and regret

We consider the full-information online setting: in round t, the decision maker must choose a solution $S_t \in \mathcal{S}$, knowing only the costs up to round $t-1$. Upon this choice, the cost f_t is revealed, and the player incurs loss $f_t(S_t)$. Throughout rounds $t = 1, \dots, T$, one aims to minimize the regret, the difference to the best fixed solution in hindsight:

$$R(T) = \frac{1}{T} \left(\sum_{t=1}^{T} f_t(S_t) - \min_{S \in \mathcal{S}} \sum_{t=1}^{T} f_t(S) \right). \quad (2)$$

An algorithm is *Hannan-consistent* if its regret vanishes as $T \to \infty$. Regret is commonly used for problems where the minimization for a known cost, $\min_{S \in \mathcal{S}} f(S)$, can be solved exactly. But problems of the form (1) with submodular costs are NP-hard; indeed, many have non-constant lower bounds on the approximation factor α. The factor α is a bound on the quality of a solution S' returned by a given algorithm, compared to the optimal solution S^*: $f(S') \leq \alpha f(S^*)$. If E are the edges in a graph $G = (V, E)$, lower bounds on α are $\Omega(|V|)$ for minimum spanning tree and perfect matching (Goel et al., 2009), $\Omega(\sqrt{|E|})$ for min (s,t)-cut (Jegelka & Bilmes, 2011b), and $\Omega(|V|)$ for edge cover (Iwata & Nagano, 2009).

Thus, in this work, we target *online approximation algorithms*. Let α be the approximation factor attained by an offline approximation algorithm that solves $\min_{S \in \mathcal{S}} f(S)$ for a known submodular f. The *α-regret* compares to the best solution that can be expected in polynomial time and is used with approximations (Kakade et al., 2009; Streeter & Golovin, 2008):

$$R_\alpha(T) = \frac{1}{T} \left(\sum_{t=1}^{T} f_t(S_t) - \alpha \min_{S \in \mathcal{S}} \sum_{t=1}^{T} f_t(S) \right). \quad (3)$$

1.2. Contributions and Roadmap

Building on offline approximation algorithms, we tackle submodular cost functions in combinatorial online problems. This setting has also been termed "learning structured concept classes" – but, contrary to previous work, we use nonlinear costs. In particular, we derive algorithms that handle (i) non-separability for structures that are beyond previous dynamic programming approaches (Lugosi et al., 2009), and (ii) approximations, by exploiting properties of submodular functions. First, we show two generic Hannan-consistent algorithms for two main approximation strategies, one based on subgradient descent (§2.1), and one based on a Follow-the-leader scheme (§2.2). Table 1 shows regret bounds with details plugged in for various problems. As a corollary, our Theorem 1 tightens Theorem 1 in (Hazan & Kale, 2009) for the unconstrained case. While the first two parts address general submodular functions, the third part focuses on a special class, namely label costs (LC). This class allows better approximation factors if class-specific algorithms are used. We reformulate LC problems as cover-type problems, and derive an

Table 1. Overview of regret bounds. The approximation factor α is underlined, k is the maximum frequency, U the universe to cover. In a graph $G = (V, E)$, $n = |V|$; for set cover, m is the number of sets. *for complete bipartite graphs

	set cover	vertex cover	(s,t)-cut	spanning tree	perfect matching
subgradient desc. (§2.1)	$O(\underline{k}\sqrt{m/T})$	$O(\underline{2}\sqrt{m/T})$	$O(\underline{n}\sqrt{m/T})$	–	–
FPL (§2.2)	–	–	$O(\underline{n}m/\sqrt{T})$	$O(\underline{n}m/\sqrt{T})$	$O(\underline{n}m/\sqrt{T})$
label cost (§2.3)	$O(\underline{\ln \lvert U\rvert}\sqrt{\lvert\mathcal{L}\rvert/T})$	$O(\underline{\ln \lvert E\rvert}\sqrt{\lvert\mathcal{L}\rvert/T})$	$O(\sqrt{\underline{m}\lvert\mathcal{L}\rvert/T})$	$O(\underline{\ln n}\sqrt{\lvert\mathcal{L}\rvert/T})$	$O(\underline{\lvert\mathcal{L}\rvert}\sqrt{\lvert\mathcal{L}\rvert/T})^*$

online algorithm that can use *any* offline algorithm for the LC problem at hand. Beyond standard label costs, our formulation extends to multiple labels and simple discounts.

1.3. Related work

Most existing online and bandit algorithms for combinatorial problems expect a modular (linear) cost function (Kalai & Vempala, 2005; Koolen et al., 2010; Cesa-Bianchi & Lugosi, 2009; Awerbuch & Kleinberg, 2004; Balcan & Blum, 2007). Many exploit the *separability* of this function to handle the exponential number of choices, e.g., when maintaining weights. Submodular functions are not separable in this way. One example for non-separable costs with multi-task constraints is (Lugosi et al., 2009). Their dynamic programming approach, however, works only for limited constraint sets that keep the state graph small.

Unconstrained submodular minimization is not NP-hard, and Hazan & Kale (2009) derive online algorithms for this problem with a regret of $O(m\sqrt{T})$ for m elements. We partially build on their techniques.

Contrary to the problems in most of the work above, most instances of Problem (1) are NP-hard. In general, there is no generic solution for integrating approximations into online algorithms. Kalai & Vempala (2005) extend the regret bound for the *Follow-the-perturbed leader* (FPL) algorithm to NP-hard combinatorial problems with a *modular* cost function if there is an algorithm that provides a coordinate-wise approximation to the optimal solution; this does not apply here. Kakade et al. (2009) show an example where FPL fails for the greedy set cover algorithm, and ask how to use FPL in general with approximations. In Section 2.2, we integrate a class of approximation algorithms for Problem (1) into the FPL framework. Kakade et al. (2009) show how to derive online approximation algorithms from offline algorithms, generalizing online gradient descent (Zinkevich, 2003) by approximate projections. They too consider only a certain family of cost functions and pose the case of *nonlinear* costs as an open problem. A straightforward use of their algorithm would yield exponential regret bounds, and also pose other problems (Supplement, 2011).

For approximate submodular maximization, online greedy methods exist (Streeter & Golovin, 2008) that satisfy constraints in expectation only. Adaptive/interactive submodularity (Golovin & Krause, 2010; Guillory & Bilmes, 2011) also implies greedy algorithms, but considers a setting different from ours.

1.4. Preliminaries and notation

The cost function f is defined on 2^E, the power set of E. The cardinality of E is $m = |E|$. Graphs are denoted by $G = (V, E)$. The α in the regret bound always refers to the approximation factor of the corresponding offline algorithm. Let $\chi_A \in \{0,1\}^E$ be the *characteristic vector* of $A \subseteq E$, which means $\chi_A(e) = 1$ if $e \in A$ and $\chi_A(e) = 0$ otherwise. An important concept for submodular functions is the *submodular polyhedron* $P_f = \{x \in \mathbb{R}^E \mid x \cdot \chi_A \leq f(A) \text{ for all } A \subseteq E\}$. For any submodular f, it holds that $f(A) = \max_{y \in P_f} y \cdot \chi_A$. The *Lovász extension* $\tilde{f}$ of f is the convex extension $\tilde{f} : \mathbb{R}^E_+ \to \mathbb{R}$ with $\tilde{f}(x) = \max_{y \in P_f} y \cdot x$, so $\tilde{f}(\chi_A) = f(A)$ for all $A \subseteq E$ (Lovász, 1983). This definition shows that $g = \operatorname{argmax}_{y \in P_f} y \cdot x$ is a subgradient of $\tilde{f}$ in x (Fujishige, 2005): it implies $g \cdot x' \leq \tilde{f}(x')$ for all $x' \in \mathbb{R}^E_+$, and hence $\tilde{f}(x') - \tilde{f}(x) \geq g \cdot (x' - x)$. The greedy algorithm (Edmonds, 1970; Lovász, 1983) finds the vector g in $O(m \log m)$ time. We will also use that the sum of submodular functions is submodular, and assume f to be nonnegative and nondecreasing. For more details on submodular functions, see e.g. (Fujishige, 2005; Lovász, 1983). Note that in a discrete space, we use "modular" and "linear" interchangeably.

2. Strategies

Many approximation algorithms relate an inherently difficult problem to an easier one, and we will build on exactly this relation. Problem (1) is hard for two reasons, and we categorize algorithms by which of those reasons they address: (i) Problem (1) has combinatorial constraints – unconstrained submodular minimization is not NP-hard (ref. in (Fujishige, 2005)); (ii) the cost function is nonseparable – for a separable f, many instances are not NP-hard, e.g., minimum spanning tree.

Algorithms working with (i) treat f as a pseudo-boolean

Algorithm 1 Rounded subgradient descent

Input: $\eta > 0$, initial $x_1 \in \mathcal{K}$
for $t = 1$ **to** T **do**
  get S_t from x_t by rounding with guarantee α
  obtain f_t
  compute $g_t = \operatorname{argmax}_{g \in P_{f_t}} g \cdot x_t$ and
  $x_{t+1} = \Pi_{\mathcal{K}}(x_t - \eta g_t)$
end for

function on indicator vectors, relax $\mathcal{S}$ to its convex hull, and finally round the solution of the relaxed problem. The relaxation is a convex non-smooth minimization problem with linear constraints. Such a problem is naturally amenable to an online, possibly exponentiated, subgradient descent.

Algorithms motivated by (ii) replace f by a tractable approximation $\hat{f}$, and minimize $\hat{f}$ over $\mathcal{S}$. We use this $\hat{f}$ in a specific way in the Follow-the-leader framework, and show example functions that fit our framework. The generic $\hat{f}$ by (Goemans et al., 2009) does not fit Algorithm 2, but we present a modification that works.

Finally, label costs permit better approximations that the generic algorithms of type (i) and (ii) are not guaranteed to achieve. Thus, we derive a third framework for label costs. It transforms the cost function and constraints into an equivalent problem – selecting labels – that is the starting point for the online approximation.

2.1. Relaxations

We begin with an algorithm that operates on a relaxation and then uses rounding. The rounding procedure determines the approximation factor; suitable procedures exist for covering constraints (Iwata & Nagano, 2009) and cuts (Jegelka & Bilmes, 2011b).

Let $\mathcal{K} \subseteq [0,1]^E$ be the convex hull of the decision space $\mathcal{S} \subseteq \{0,1\}^E$; both are described by the same linear inequalities. The cost function on $\mathcal{K}$ corresponding to f_t on $\mathcal{S}$ is the convex Lovász extension $\tilde{f}_t$. Algorithm 1 maintains two variables: it performs a subgradient descent based on (Zinkevich, 2003) in continuous space that yields x_t, and then rounds to S_t. In each round t, it takes a step into the direction of the negative subgradient $-g_t$ of $\tilde{f}_t$ and projects back onto $\mathcal{K}$. The projection $\Pi_{\mathcal{K}}(y) = \operatorname{argmin}_{x \in \mathcal{K}} \|x - y\|^2$ is in general easier to solve than the full non-smooth relaxation.

Theorem 1. *For a rounding scheme with approximation guarantee α, $M = \max_t f_t(E)$, and $\eta = \sqrt{m}(M\sqrt{T})^{-1}$, Algorithm 1 has an α-regret of $R_\alpha(T) \leq \alpha M \sqrt{m/T} = O(\alpha\sqrt{m/T})$.*

As a corollary, Theorem 1 improves a bound in (Hazan & Kale, 2009), and makes it tight (using $\alpha = 1$, $\mathcal{S} = \{0,1\}^E$ and their thresholded rounding).

Corollary 1. *The regret for online submodular minimization with Algorithm 1 is bounded by $O(\sqrt{m/T})$.*

Crucial for the improvement is a bound on the norm of the subgradient that uses $g_t \in P_f$ and $\|g_t\| \leq \|g_t\|_1$ (Supplement, 2011):

Lemma 1. *Let g_t be a subgradient of f_t (obtained as in §1.4). Then $\|g_t\| \leq \beta \max_{A \subseteq E} |f_t(A)| - f_t(\emptyset)$, where $\beta = 1$ if f_t is nondecreasing, and $\beta = 3$ otherwise.*

Proof. (Thm. 1, outline) The proof consists of two steps. First, we bound the 1-regret for the sequence $\{x_t\}$ analogous to (Zinkevich, 2003), and then use this result to bound the α-regret for the sequence S_t. Let $S^* \in \operatorname{argmin}_{S \in \mathcal{S}} \sum_{t=1}^T f_t(S)$. The definition $\tilde{f}_t(x) = \max_{g \in P_{f_t}} g \cdot x$ implies $f(S^*) = \tilde{f}(\chi_{S^*})$ and

$$\sum_{t=1}^T \tilde{f}(x_t) - \sum_{t=1}^T f(S^*) \leq \sum_{t=1}^T g_t \cdot x_t - \sum_{t=1}^T g_t \cdot \chi_{S^*}.$$

A proof similar to that in (Zinkevich, 2003) leads to a bound on the right hand side that we bound further:

$$\begin{aligned} 2\sum_{t=1}^T g_t \cdot (x_t - \chi_{S^*}) &\leq \max_{x,y \in \mathcal{K}} \|x - y\|^2/\eta + \eta T \max_t \|g_t\|^2 \\ &\leq m/\eta + M^2 T \eta. \end{aligned}$$

For the second inequality, we used $\|x - y\|^2 \leq m$ for all $x, y \in \mathcal{K}$ because $\mathcal{K} \subseteq [0,1]^E$. Furthermore, we bounded the ℓ_2 norm of g_t by Lemma 1:

$$\|g_t\| \leq g_t \cdot \chi_E \leq \max_t f_t(E) \leq M.$$

Finally, the approximation guarantee for the rounding procedure implies that $f(S_t) = \tilde{f}(\chi_{S_t}) \leq \alpha \tilde{f}(x_t)$, so

$$\begin{aligned} \sum_{t=1}^T f(S_t) - \alpha \sum_{t=1}^T f(S^*) &\leq \alpha \sum_{t=1}^T \tilde{f}(x_t) - \alpha \sum_{t=1}^T f(S^*) \\ &\leq 0.5\alpha(m/\eta + M^2 T \eta). \end{aligned}$$

The regret bound follows with $\eta = \sqrt{m}(M\sqrt{T})^{-1}$. □

A similar strategy works with exponentiated gradients – the regret bounds are as for linear cost problems, scaled by a factor α. The proof is analogous.

Suitable rounding techniques are not available for all submodular-cost problems. Instead, several algorithms approximate the cost function. This approach fits the Follow-the-leader framework described next.

2.2. Approximations of the cost

We now address algorithms of type (ii) that replace f in Problem (1) by an approximation $\hat{f}$ and solve the resulting tractable problem instead. Several $\hat{f}$ integrate with the Follow-the-leader principle (Hannan, 1957).

This principle suggests playing the best S given the costs observed so far: in round t, pick $S_t = \operatorname{argmin}_{S\in\mathcal{S}} \sum_{\tau=1}^{t-1} f_\tau(S)+r(S)$ (Kalai & Vempala, 2005). In the simplest case, the last term is a modular perturbation $r(S) = r \cdot \chi_S$ by a random vector r (Follow-the-perturbed-leader, FPL). But here, finding such a minimizer S_t is NP-hard, and an S_t from an approximation algorithm is not enough (Kakade et al., 2009). Instead, Algorithm 2 uses the *exact* expected minimizer of the *approximate* costs, because the deviation of *each* $\hat{f}_t$ from f_t is bounded, not only the sum. We define two general conditions for $\hat{f}$:

C1 The approximation $\hat{f}$ of f satisfies $f(A) \le \hat{f}(A) \le \alpha f(A)$ for all $A \in \mathcal{S}$.

C2 The following problem can be solved exactly in polynomial time:

$$\operatorname{argmin}_{S\in\mathcal{S}} \sum\nolimits_t \hat{f}_t(S) + \alpha r(S). \tag{4}$$

Algorithm 2 Follow the approx. perturbed leader

Input: $\eta > 0$
pick $r \in [0, M/\eta]^E$ uniformly at random
for $t = 1$ **to** T **do**
  approximate f_t by $\hat{f}_t$
  set $S_t = \operatorname{argmin}_{S\in\mathcal{S}} \sum_{\tau=1}^{t-1} \hat{f}_\tau(S) + \alpha r(S)$
  obtain f_t
end for

These constraints apply in a variety of settings, as we discuss later. Algorithm 2 integrates such an $\hat{f}$ into the FPL framework and adapts the perturbation r accordingly. Condition (C2) ensures that we can find S_t. Then we can bound the regret for submodular costs in an approximation setting even with FPL.

Theorem 2. *For an approximation $\hat{f}$ that satisfies (C1) and (C2), $M = \max_t f_t(E)$, and $\eta = 1/\sqrt{2T}$, Algorithm 2 achieves an expected α-regret $\mathbb{E}[R_\alpha(T)] \le 2\sqrt{2}\alpha m M/\sqrt{T} = O(\alpha m/\sqrt{T})$.*

Proof. (Outline) Let

$$S_t = \operatorname*{argmin}_{S\in\mathcal{S}} \sum\nolimits_{\tau=1}^{t-1} \hat{f}_\tau(S) + \alpha r(S);$$

$$\widehat{S}_t = \operatorname*{argmin}_{S\in\mathcal{S}} \sum\nolimits_{\tau=1}^{t-1} \hat{f}_\tau(S); \quad S_t^* = \operatorname*{argmin}_{S\in\mathcal{S}} \sum\nolimits_{\tau=1}^{t} f_\tau(S).$$

Similar to the proof of Lemma 3.1 in (Kalai & Vempala, 2005), we obtain a bound for the series of S_{t+1}:

$$\sum\nolimits_{t=1}^{T} \hat{f}_t(S_{t+1}) \le \sum\nolimits_{t=1}^{T} \hat{f}_t(\widehat{S}_{T+1}) + \alpha(r(\widehat{S}_{T+1}) - r(S_1)).$$

To transfer this result to the series of S_t, we use that $\hat{f}_t(S_t) \le \hat{f}_t(S_{t+1}) + (\hat{f}_t(S_t) - \hat{f}_t(S_{t+1}))$:

$$\begin{aligned}\sum_{t=1}^{T} \hat{f}_t(S_t) \le{}& \sum_{t=1}^{T} \hat{f}_t(\widehat{S}_{T+1}) \\ &+ \sum_{t=1}^{T} (\hat{f}_t(S_t) - \hat{f}_t(S_{t+1})) + \alpha(r(\widehat{S}_{T+1}) - r(S_1)).\end{aligned} \tag{5}$$

Condition (C1) implies that

$$\sum\nolimits_{t=1}^{T} \hat{f}_t(\widehat{S}_{T+1}) \le \sum\nolimits_{t=1}^{T} \hat{f}_t(S_T^*) \le \alpha \sum\nolimits_{t=1}^{T} f_t(S_T^*),$$

and that $\sum_{t=1}^{T} f_t(S_t) \le \sum_{t=1}^{T} \hat{f}_t(S_t)$. Together with Equation (5), this yields

$$\begin{aligned}&\sum\nolimits_{t=1}^{T} f_t(S_t) - \alpha \sum\nolimits_{t=1}^{T} f_t(S_T^*) \\ &\quad\le \sum\nolimits_{t=1}^{T} (\hat{f}_t(S_t) - \hat{f}_t(S_{t+1})) + \alpha(r(\widehat{S}_{T+1}) - r(S_1)).\end{aligned}$$

It remains to bound the two terms on the right hand side, and these bounds depend on $r \in [0, M/\eta]^E$. The expected difference of r terms can be bounded by mM/η. For the other term, we extend a technique in (Hazan & Kale, 2009) to the approximation setting (Supplement, 2011) to bound the probability that $S_t \ne S_{t+1}$, and reach the inequality

$$\begin{aligned}&\sum\nolimits_{t=1}^{T} \mathbb{E}[\hat{f}_t(S_t) - \hat{f}_t(S_{t+1})] \\ &\quad\le \sum\nolimits_{t=1}^{T} P(S_t \ne S_{t+1}) \max_{A\in\mathcal{S}} \hat{f}(A) \;\le\; 2\alpha m M T \eta.\end{aligned}$$

Finally, combining these bounds yields

$$\mathbb{E}\Big[\sum_{t=1}^{T} f_t(S_t)\Big] - \alpha \sum_{t=1}^{T} f_t(S_T^*) \le \alpha M m/\eta + 2\alpha m M T \eta.$$

Theorem 2 follows for $\eta = 1/\sqrt{2T}$. □

2.2.1. Approximations fitting Algorithm 2

Conditions (C1) and (C2) show that a suitable approximation $\hat{f}$ is decisive. We list and derive examples for $\hat{f}$ that can be plugged into Algorithm 2.

Spanning tree and matching. The best approximation bound for minimum spanning tree (MST) and perfect matching with general submodular costs is $O(|V|)$, and is achieved with the simple approximation $\hat{f}_m(S) = \sum_{e\in S} f(e)$ (Goel et al., 2009). Since

$\hat{f}_m$ is additive, any standard algorithm for MST or matching applies for (C2). (C1) holds by subadditivity of f.

The simple $\hat{f}_m$, however, often leads to rather loose approximation factors. We derive a better, nontrivial approximation for the problem structure of cuts.

Minimum cut. For cuts, E is the set of directed edges in a graph $G = (V, E)$. We design $\hat{f}$ to be separable across certain groups of edges for tractability, but retain submodularity on restricted sets to improve on $\hat{f}_m$. Let $\{E_v^-\}_{v\in V}$ be the partition of E where E_v^- contains all edges $e = (u, v)$ with head v, and $\{E_u^+\}_{u\in V}$ be the analogous partition that assigns each edge $e = (u, v)$ to its tail node u. For either partition, we define an approximate cost function:

$$\hat{f}^-(S) = \sum_{v\in V} f(A \cap E_v^-); \quad \hat{f}^+(S) = \sum_{v\in V} f(A \cap E_v^+).$$

By subadditivity, both functions are upper bounds on f. Before starting Algorithm 2, we decide uniformly at random whether to use $\hat{f} = \hat{f}^-$ or $\hat{f} = \hat{f}^+$ and retain this choice throughout. With this strategy, the factor α in (C1) improves from m (for $\hat{f}_m$) to $|V|/2$:

Lemma 2. *Let $\hat{f}$ be randomly chosen between $\hat{f}^-$ and $\hat{f}^+$ with equal probabilities. Then $f(S) \le \mathbb{E}[\hat{f}(S)] \le (|V|/2)f(S)$ for all minimal (s,t)-cuts S.*

The lemma follows from the definition of $\hat{f}$ and subadditivity of f (Supplement, 2011).

To satisfy (C2), we compute the cut $\mathcal{S}_t$ via a generalized flow problem. Polymatroidal network flows (Lawler & Martel, 1982) generalize flow capacity constraints to submodular functions $\mathrm{cap}_v^{\mathrm{in}}$ and $\mathrm{cap}_v^{\mathrm{out}}$ that restrict the inflow and outflow of each node v, respectively. Such a maxflow problem can be solved in polynomial time. Its dual problem is a minimum cut with cut cost $c(S) = \min_{A\subseteq S} \sum_v \mathrm{cap}_v^{\mathrm{in}}(A\cap E_v^-) + \mathrm{cap}_v^{\mathrm{out}}((S\setminus A)\cap E_v^+)$ (Lovász, 1983). We thus set the capacity functions to represent the cost in (C2). If $\hat{f} = \hat{f}^-$, then we set $\mathrm{cap}_v^{\mathrm{out}}$ to some large value so that $c(S)$ only uses $\mathrm{cap}_v^{\mathrm{in}}$, and set $\mathrm{cap}_v^{\mathrm{in}}(A) = \sum_t f_t(A \cap E_v^-) + \alpha r(A \cap E_v^-)$. The procedure for $\hat{f}^+$ is analogous. Then (here for $\hat{f}^-$)

$$\begin{aligned} c(S) &= \sum_v \sum_t f_t(S \cap E_v^-) + \alpha r(S \cap E_v^-) \\ &= \sum_t \hat{f}_t(S) + \alpha r(S). \end{aligned}$$

2.2.2. A generic approximation

A generic nontrivial approximation $\hat{f}$ for submodular functions was proposed by Goemans et al. (2009), but it does not satisfy (C2) in a straightforward way. Its functional form is $\hat{f}(S) = \sqrt{\sum_{e\in S} w(e)}$.

Nevertheless, it is possible to use its square. It satisfies $\hat{f}^2(A) \le f^2(A) \le \alpha_g^2 \hat{f}^2(A)$ for all $A \subseteq E$, with $\alpha_g = O(\sqrt{m} \log m)$. Even better, $\hat{f}^2$ is a modular function, so we can use any online algorithm for linear costs: when observing f_t, we pretend to have seen $\hat{f}_t^2$. The regret might worsen by a constant factor, compared to a linear loss of the same range as f_t.

To state the regret bound, let $\nu = \min_{t,S\in\mathcal{S}} f_t(S)$. We make the reasonable assumption that no f_t is the constant zero function, and then $\nu > 0$. The following Lemma is proved in (Supplement, 2011).

Lemma 3. *Let $\widehat{R}_{\mathcal{A}}$ be the regret of an online algorithm $\mathcal{A}$ when used with linear cost functions with a range like $\hat{f}_t^2$. Using $\mathcal{A}$ with $\hat{f}_t^2$ when observing f_t leads to an α_g-regret of $R_{\alpha_g}(T) \le \alpha_g \widehat{R}_{\mathcal{A}}/\nu$.*

2.3. Labels and related costs

The two previous sections proposed online algorithms for general submodular functions. The associated approximation factors usually match their lower bounds. However, certain sub-classes of submodular functions admit better approximation factors. As an example, the approximation factor for MST drops from linear to logarithmic in $|V|$ if f is a label cost function. Therefore, we address a specific algorithm for label costs.

With label costs, each element e has a label $\pi(e) \in \mathcal{L}$, and the cost is the additive cost of the labels, $f(S) = c(\bigcup_{e\in S} \pi(e)) = \sum_{\ell\in\pi(S)} c(\ell)$ (Hassin et al., 2007). We assume here that the labels of the elements are fixed, but the cost of the labels changes over time. We start with trees and covers, and then generalize our technique to related costs and other problems.

We intrinsically transform the decision space and state the problem of selecting a structure as a label selection problem with a covering constraint. Then we exploit the simpler cost on the labels. The offline approximation algorithm helps find the desired labels.

Instead of directly choosing a structure, Algorithm 3 picks a set of *labels* L_t with minimum cost. This L_t must be such that the set of elements $E(L_t) = \{e|\pi(e) \in L_t\}$ with labels in L_t contains the desired structure $S_t \in \mathcal{S}$, e.g., a tree. Given such an L_t, finding a feasible $S_t \subseteq E(L_t)$ is easy, and, by the definition of f_t, $f_t(S_t) \le c_t(L_t)$. For trees, we find S_t by pruning the graph to contain only edges $E(L_t)$ and then compute a spanning tree, S_t.

It remains to determine (i) by which criterion to choose L_t, and (ii) how to find an L_t that contains the desired structure. As to (i), we note that the cost function on labels is modular. Therefore, we use an approximate

Algorithm 3 Online label cost minimization

pick any $S_1 \in \mathcal{S}$; set $L_1 = \bigcup_{e \in S} \pi(e)$, $y_1 = \chi_{L_1}$
for $t = 2$ **to** T **do**
  $(y_t, L_t) = \text{ApproxProj}(y_{t-1} - \eta c_{t-1}, L_{t-1}, y_{t-1})$
  find $S_t \subseteq E(L_t)$, $S_t \in \mathcal{S}$
  obtain f_t and extract c_t
end for

gradient descent. Algorithm 3 maintains a continuous correspondent y_t of L_t, and moves from y_t into the direction of the negative gradient c_t. To move the resulting point to the feasible set, we use an approximate projection from (Kakade et al., 2009), denoted by ApproxProj. This method applies because our re-formulation has linear cost. The approximate projection relies on an approximation algorithm - here, an algorithm to find a suitable L_t, given a cost vector c.

Thus, as a last step, we show how to find a label set L_t that contains a spanning tree. Given cost c, we formulate a submodular cover problem:

$$\min\ c(L) \quad \text{s.t. } g(L) = g(\mathcal{L}), \tag{6}$$

for a nondecreasing submodular function $g : 2^{\mathcal{L}} \to \mathbb{N}_0$. We construct g to capture the tree constraint for $E(L)$. Let $r : 2^E \to \mathbb{N}_0$ be the rank function of a graphic matroid defined by the given graph, that is, $r(S) = |S|$ if the subgraph induced by S does not contain any cycles. Otherwise, $r(S)$ is the size of the largest subset of S that is cycle-free. The function r is submodular and nondecreasing. If S is a spanning tree, then $r(S) = r(E) = |V| - 1$. We define

$$g(L) = r(E(L)). \tag{7}$$

This function g is also submodular, integral and nondecreasing, and $g(L) = g(\mathcal{L}) = r(E)$ if and only if $E(L)$ contains a spanning tree. Problem (6) is solved by a greedy algorithm for submodular cover (Wolsey, 1982), with approximation factor $\alpha = H(\max_\ell g(\ell)) = O(\log |V|)$. Here, $H(n)$ is the nth harmonic number. This algorithm completes the online algorithm for spanning trees.

The construction for set cover is analogous, with $g(L)$ counting the number of elements covered by sets with labels in L. In fact, the result is a standard set cover problem. Thanks to our reformulation, the regret bound involves the total number of labels, $|\mathcal{L}| \leq m$.

Theorem 3. *The regret of Algorithm 3 is bounded as* $R_\alpha(T) = O(\alpha\sqrt{M|\mathcal{L}|/T})$*, where* $M = \max_t f_t(E)$.

The bound follows from Theorem 3.2 in (Kakade et al., 2009) for $\eta = (\alpha + 1)\sqrt{|\mathcal{L}|/(MT)}$, and from the equivalence of picking labels and structures, as outlined above. Theorem 3 immediately implies a regret bound of $O(\sqrt{|\mathcal{L}|M} \log |V|/\sqrt{T})$ for spanning trees.

Multiple labels and truncated costs. Our transformation also applies to multiple labels per edge and simple thresholded costs. Multiple labels can be simulated by dividing an edge into "slots", and by a construction to count edges fractionally. Details are described in (Supplement, 2011). In a similar spirit, truncated costs of the form $c(L) = \min\{w \cdot \chi_L, \gamma\}$ can be simulated by parallel edges, so that the algorithm can pick a full group or single edges via labels.

Other structures. Some structures, such as paths, are not easily represented as submodular covers. However, it actually suffices to view the label cost problems as modular-cost problems of choosing a minimum cost set of labels L, where $E(L)$ must contain the desired structure. To solve this problem, we compute a minimum label cost structure S^* and pick all labels used by S^*. Algorithms for minimum label cost path (Hassin et al., 2007), matching (Monnot, 2005) or cut (Zhang et al., 2009) find an approximate S^* with factor α. We substitute the respective procedure for the submodular cover in the approximate projections, and Algorithm 3 still applies. As a result, Theorem 3 holds with the respective α.

3. Discussion and Open Questions

We showed two generic approaches to online algorithms for submodular costs and combinatorial structures: projected subgradient for relaxations and Follow-the-leader for approximations of the cost function. Despite being not completely ignorant of the underlying approximation strategy, they are still generic enough to cover almost all existing offline approximation methods for structured concepts with submodular costs. Algorithm 3 is even more generic: it fits *any* algorithm for a label cost problem, and yields vanishing α-regret even for the better α attainable with label costs. Details for all results are in (Supplement, 2011).

Our work contributes to extending the results for the online combinatorial setting from linear costs to nonlinear costs. In particular, it yields the first regret bounds for structured concepts with submodular costs. This complements the work in the unconstrained setting (Hazan & Kale, 2009), and the work on online submodular maximization (Streeter & Golovin, 2008; Streeter et al., 2009). "Combinatorial bandits" have been explored for linear cost functions (Cesa-Bianchi & Lugosi, 2009) – an open question remains what is achievable in the bandit case for nonlinear, nonseparable costs.

References

Atamtürk, A. and Narayanan, V. Polymatroids and mean-risk minimization in discrete optimization. *Operations Research Letters*, 36(5):618–622, 2008.

Awerbuch, B. and Kleinberg, R. D. Adaptive routing with end-to-end feedback: distributed learning and geometric approaches. In *Proc. of the ACM Symp. on Theory of Computing (STOC)*, 2004.

Balcan, M. F. and Blum, A. Approximation algorithms and online mechanisms for item pricing. *Theory of Computing*, 3(9):179–195, 2007.

Calinescu, G., Kapoor, S., Olshevsky, A., and Zelikovsky, A. Network lifetime and power assignment in ad hoc wireless networks. In *Proc. of the European Symp. on Algorithms (ESA)*, 2003.

Cesa-Bianchi, N. and Lugosi, G. Combinatorial bandits. In *Proc. of the Int. Conf. on Learning Theory (COLT)*, 2009.

Edmonds, J. *Combinatorial Structures and their Applications*, chapter Submodular functions, matroids and certain polyhedra, pp. 69–87. Gordon and Breach, 1970.

Fujishige, S. *Submodular Functions and Optimization*. Number 58 in Annals of Discrete Mathematics. Elsevier Science, 2nd edition, 2005.

Goel, G., Karande, C., Tripati, P., and Wang, L. Approximability of combinatorial problems with multi-agent submodular cost functions. In *Proc. of the Ann. Symp. on Foundations of Comp. Science (FOCS)*, 2009.

Goemans, M. X., Harvey, N. J. A., Iwata, A., and Mirrokni, V. S. Approximating submodular functions everywhere. In *Proc. of the ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2009.

Golovin, D. and Krause, A. Adaptive submodularity: A new approach to active learning and stochastic optimization. In *Proc. of the Int. Conf. on Learning Theory (COLT)*, 2010.

Guillory, A. and Bilmes, J. Simultaneous learning and covering with adversarial noise. In *Int. Conference on Machine Learning (ICML)*, Bellevue, Washington, 2011.

Hannan, J. Approximation to Bayes risk in repeated play. In *Contr. to the Theory of Games*, volume III. 1957.

Hassin, R., Monnot, J., and Segev, D. Approximation algorithms and hardness results for labeled connectivity problems. *Journal of Combinatorial Optimization*, 14(4): 437–453, 2007.

Hazan, E. and Kale, S. Online submodular minimization. In *Proc. of the Ann. Conf. on Neural Info. Processing Systems (NIPS)*, 2009.

Iwata, S. and Nagano, K. Submodular function minimization under covering constraints. In *Proc. of the Ann. Symp. on Foundations of Comp. Science (FOCS)*, 2009.

Jegelka, S. and Bilmes, J. Submodularity beyond submodular energies: coupling edges in graph cuts. In *Proc. of the IEEE Conf. on Computer Vision and Pattern recognition (CVPR)*, 2011a.

Jegelka, S. and Bilmes, J. Approximate inference via generalized graph cuts. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2011b.

Jha, S., Sheyner, O., and Wing, J.M. Two formal analyses of attack graphs. In *Proc. of the 15th Computer Security Foundations Workshop*, pp. 49–63, 2002.

Kakade, S., Kalai, A. T., and Ligett, K. Playing games with approximation algorithms. *SIAM Journal on Computing*, 39(3):1088–1106, 2009.

Kalai, A.T. and Vempala, S. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71:26–40, 2005.

Koolen, W.M., Warmuth, M. K., and Kivinen, J. Hedging structured concepts. In *Proc. of the Int. Conf. on Learning Theory (COLT)*, 2010.

Koufogiannakis, C. and Young, N. E. Greedy Δ-approximation algorithm for covering with arbitrary constraints and submodular costs. In *Proc. of the Int. Colloquium on Automata, Languages and Programming (ICALP)*, 2009.

Lawler, E. L. and Martel, C. U. Computing maximal "Polymatroidal" network flows. *Mathematics of Operations Research*, 7(3):334–347, 1982.

Lovász, L. *Mathematical programming – The State of the Art*, chapter Submodular Functions and Convexity, pp. 235–257. Springer, 1983.

Lugosi, G., Papaspiliopoulos, O., and Stoltz, G. Online multi-task learning with hard constraints. In *Proc. of the Int. Conf. on Learning Theory (COLT)*, 2009.

Monnot, J. The labeled perfect matching in bipartite graphs. *Information Processing Letters*, 96:81–88, 2005.

Streeter, M. and Golovin, D. An online algorithm for maximizing submodular functions. In *Proc. of the Ann. Conf. on Neural Info. Processing Systems (NIPS)*, 2008.

Streeter, M., Golovin, D., and Krause, A. Online learning of assignments. In *Proc. of the Ann. Conf. on Neural Info. Processing Systems (NIPS)*, 2009.

Supplement. `http://ssli.ee.washington.edu/~jegelka/icml/online.pdf`, 2011.

Wolsey, L. An analysis of the greedy set cover algorithm for the submodular set covering problem. *Combinatorica*, 2(4), 1982.

Yuan, S., Varma, S., and Jue, J. P. Minimum-color path problems for reliability in mesh networks. In *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2005.

Zhang, P., J.-Y, Cai, Tang, L.-Q., and Zhao, W.-B. Approximation and hardness results for label cut and related problems. *Journal of Combinatorial Optimization*, 2009.

Zinkevich, M. Online convex programming and infinitesimal gradient ascent. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2003.