Online algorithms for submodular minimization with combinatorial constraints

Stefanie Jegelka
Max Planck Institute for Biological Cybernetics
Tübingen, Germany
jegelka@tuebingen.mpg.de

Jeff Bilmes
University of Washinton
Seattle, USA
bilmes@uw.edu

Abstract

Building on recent results for submodular minimization with combinatorial constraints, and on online submodular minimization, we address online approximation algorithms for submodular minimization with combinatorial constraints. We discuss two types of algorithms and outline approximation algorithms that integrate into those.

1 Introduction

Theoretical computer science has seen a rising interest in submodular minimization problems subject to combinatorial constraints [5, 10, 14, 15, 16, 27]. Such problems result from replacing the linear (modular) cost function in a combinatorial optimization problem by a submodular function. Formally, we are given a set E of elements, e.g., the edges in a graph, and a set of feasible solutions, defined as a family $\mathcal{C} \subseteq 2^E$ of subsets of E. These sets may be, for instance, all spanning trees in the graph, all (s,t)-paths, or all cuts. For a cost function $f : 2^E \to \mathbb{R}_+$, the optimization problem is

$$\min\{f(S) \mid S \in \mathcal{C}\}. \tag{1}$$

In the standard case, there are nonnegative weights $w : E \to \mathbb{R}_+$, and f is the sum of weights: $f(S) = \sum_{e \in S} w(e)$. This leads to the shortest path problem, minimum (s,t)-cut, and so on.

In this work, f is a non-decreasing *submodular* set function. This generalization was already encouraged in [21]. A function is submodular if it satisfies diminishing marginal costs: for any $A \subset B \subseteq E \setminus e$, it holds that $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$. The sum of weights satisfies this with equality, i.e., it is a *modular* set function. The minimum cut problem with submodular cost, for instance, has applications in the analysis of attack graphs in computer security, or in image segmentation [17]. In general, submodularity has been important in combinatorics and game theory.

In parallel to the progress in algorithms and complexity, submodularity has emerged as a useful property in Machine Learning. Recently, Hazan and Kale [12] developed algorithms for online and bandit submodular minimization without constraints ($\mathcal{C} = 2^E$). Furthermore, there are online algorithms for submodular maximization [25, 26]. The aim of this work is to explore routes for online submodular minimization with combinatorial constraints.

We consider the full-information case: in round t, we must choose a solution $S_t \in \mathcal{C}$ and then incur loss $f_t(S_t)$. After playing, we also receive the function f_t. Throughout rounds $t = 1, \ldots, T$, we aim to minimize the regret, the difference to the best fixed solution in hindsight:

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

Regret is commonly used for problems where the minimization for a known cost, $\min_{S \in \mathcal{C}} f(S)$, can be solved exactly. But problems of the form (1) with submodular costs are mostly not only NP-hard;

many have high non-constant lower bounds on the approximation factor. The approximation factor α is a bound on the quality of a solution S' returned by a given algorithm, compared to the optimal solution S^*: $f(S')/f(S^*) \leq \alpha$. For several balanced partition problems, $\alpha = o(\sqrt{|E|/\log|E|})$ [27]. For E being the edges in a graph $G = (V, E)$, minimum spanning tree and perfect matching have lower bounds of $|V|^{1-\epsilon}$, and shortest path of $|V|^{2/3-\epsilon}$ [10] for any fixed $\epsilon > 0$; and for minimum (s, t)-cut, we proved that $\alpha = o(\sqrt{|E|/\log|E|})$. Edge cover constraints make the order of α at least $\sqrt{|V|}/\log^2|V|$ [15]. That said, the cited works also contain approximation algorithms, and approximation algorithms can work very well in practice.

Thus, in this work, we are searching for *online approximation* algorithms. Let α be the approximation factor attained by an offline approximation algorithm that solves $\min_{S\in\mathcal{C}} f(S)$ for a known submodular f. The α-*regret* compares to the best solution that can be expected in polynomial time and is commonly used with approximations:

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

Existing approaches and difficulties. In machine learning, most existing online and bandit algorithms for combinatorial problems focus on problems that are not NP-hard. In particular, most of them expect a modular cost function $f(S) = \sum_{e\in S} w(e)$ [1, 2, 3, 4, 6, 13, 19]. They exploit the *separability* of this function. Submodular functions, however, are not separable in this way. One example for non-modular costs and multi-task constraints is [22]. They use multiplicative updates to achieve an $O(m\sqrt{mT})$ regret for m tasks. The algorithms in [12] for unconstrained submodular minimization yield a regret of $O(m\sqrt{T})$ for m elements.

In general, there is no generic solution for integrating approximations into online algorithms [18, 19]. Kalai and Vempala [19] 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 *point-wise approximation* S' to the optimal S^*, i.e., $\mathbb{E}[\chi_{S'}(e)] \leq \alpha\chi_{S^*}(e)$ for all $e \in E$. Here, χ_S denotes the characteristic vector of S. Again, non-separability renders this result inapplicable to submodular functions. Similarly, Kakade et al. [18] show an example where FPL fails for the greedy Set Cover algorithm, and ask how to use FPL in general with approximations. It is the structure of the approximation algorithm that counts. In Section 3, we integrate a class of approximation algorithms for Problem (1) into the FPL framework.

Kakade et al. [18] show how to derive online approximation algorithms from offline algorithms, generalizing online gradient descent [28] with approximate projections. They too consider only a certain family of cost functions and pose the case of *nonlinear* cost functions as an open problem. Their cost function is of the form $c : 2^E \times \mathbb{R}^d \to \mathbb{R}$, $c(S, w) = \langle\phi(S), w\rangle$ and must be *linear* in w. To use this framework, we must express any non-decreasing submodular f via its parameter w^f as $c(S, w^f) = f(S)$. The set of non-decreasing submodular functions on E is equivalent to a convex cone in $\mathbb{R}^{2^{|E|}}$. This set has a non-empty relative interior (e.g., $f(S) = \log(1 + |S|)$). As a result, simple linear algebra shows that a full basis is needed to represent all such f meaning that w has an exponential dimension d. But then the regret bound in [18] is exponential in $|E|$, since it is linear in $\|w\|$, i.e., proportional to $\sqrt{d}$. Whilst the norm issue can possibly be resolved, the algorithm also assumes that, given any $w \in \mathbb{R}^d$, we can project it onto the set of those w for which $c(\cdot, w)$ is a nondecreasing submodular function. Given the results in [23], this too seems to be non-trivial. Thus, we take a different route.

Strategies. We exploit that several approximation algorithms intrinsically relate a difficult problem to one that can be solved exactly. Problem (1) is hard for two reasons: (i) there are combinatorial constraints (unconstrained submodular minimization is not NP-hard (ref. in [8])); and (ii) the cost function is nonseparable (for a modular f, many instances of (1) are in P, such as shortest path or minimum cut). Algorithms that address (i) usually consider f as a pseudo-boolean function on indicator vectors, relax $\mathcal{C}$ to its convex hull, and finally round the solution of the relaxed problem. For submodular costs, the relaxation is a convex non-smooth minimization problem with linear constraints. Such an approach is amenable to the online (sub-)gradient descent in [28]. Algorithms motivated by (ii) replace f by a tractable approximation $\hat{f}$, and minimize $\hat{f}$ over $\mathcal{C}$. We define two

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

Algorithm 1: Rounded subgradient descent

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;

Algorithm 2: Follow the approx. perturbed leader

conditions on $\hat{f}$ under which it integrates into the FPL framework. Of course, an essential condition is to have such suitable offline approximation algorithms for the problem at hand.

1.1 Preliminaries

Let $\chi_A \in \{0,1\}^E$ be the *characteristic vector* of $S \subseteq E$, that means $\chi_A(e) = 1$ if $e \in A$ and $\chi_A(e) = 0$ otherwise. An important concept for submodular functions has been 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 $A \subseteq E$ [21]. This definition shows that $g = \operatorname{argmax}_{y \in P_f} y \cdot x$ is a subgradient of $\tilde{f}$ in x [9]: 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 [7, 21] finds the vector g in $O(m \log m)$ time. Here and in the sequel, $m = |E|$. For more details on submodular functions, see e.g. [9, 21].

2 Convex relaxations and subgradient descent

To relax the constraints in Problem (1), we view f as a pseudo-boolean function from $\{0,1\}^E$ to $\mathbb{R}_+$. The convex Lovász extension $\tilde{f}$ extends this function to the domain $[0,1]^E$. By describing $\mathcal{C}$ by linear inequalities we reach a convex integer program equivalent to (1):

$$\min \ \{\tilde{f}(x) \mid Ax \leq b \text{ and } x \in \ \{0,1\}^E\}. \tag{4}$$

Now we relax the integer constraints to $x \in [0,1]^E$. The the feasible region becomes the convex hull $\mathcal{K}$ of $\mathcal{C}$. The relaxed problem can be solved by the ellipsoid method, or, more efficiently, any method for non-smooth convex optimization. As an example, an approach such as [5] is feasible and increases the approximation factor by a factor of $(1 + \epsilon)$.

Decisive for approximation guarantees is the rounding from a fractional optimal solution x^* of the relaxed problem to an integral $\chi_S \in \mathcal{C}$. For covering constraints, Iwata and Nagano [15] prove approximation factors achieved by thresholded rounding. A related rounding procedure is possible for cuts and yields a factor $\alpha = P_{\max}$, where $P_{\max} \leq |V|$ is the longest (s,t)-path in the graph [16].

Algorithm 1 outlines the subgradient descent based on [28]. In each round t, it takes a step into the direction of the negative subgradient $-g$ and projects back onto the feasible set $\mathcal{K}$. 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. Since the norm of differences is not monotone increasing, it is necessary to prune the projection for cuts. However, this can be done easily by one modular-cost minimum cut [16]. By comparison, [18] use the full approximation algorithm to project.

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

Proof. (outline) The proof consists of two steps. First, we bound the 1-regret for the sequence $\{x_t\}$ analogous to [28], and then use this result to bound the α-regret for the sequence S_t.

Let $S^* \in \operatorname{argmin}_{S \in \mathcal{C}} \sum_{t=1}^T f_t(S)$. The definition $\tilde{f}_t(x) = \max_{g \in P_{f_t}} g \cdot x$ implies that

$$\sum\nolimits_{t=1}^T \tilde{f}(x_t) - \sum\nolimits_{t=1}^T f(S^*) \leq \sum\nolimits_{t=1}^T g_t \cdot x_t - \sum\nolimits_{t=1}^T g_t \cdot \chi_{S^*}. \tag{5}$$

Following the proof in [28] leads to a bound on the right hand side that we bound further:

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

For the second inequality, we use that $\|x-y\|^2 \leq m$ for all $x, y \in \mathcal{K}$ because $\mathcal{K} \subseteq [0,1]^m$. Furthermore, we bound the ℓ_2 norm of g_t by its ℓ_1 norm. Since $g_t \geq 0$ for a non-decreasing f_t by construction, it holds that $\|g_t\|_1 = g_t \cdot 1_m = g_t \cdot \chi_E$. Now we exploit that $g_t \in P_{f_t}$ for all t, and thus, by definition of P_f, it holds that $g_t \cdot \chi_E \leq f_t(E)$. As a result, $\|g_t\| \leq \|g_t\|_1 = g_t \cdot \chi_E \leq \max_t f_t(E) \leq M$.

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

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

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

3 Follow the approximate perturbed leader

Algorithms based on Follow-the-leader have been popular for the online setting. The principle is, in round t, to play the regularized expected leader given the functions observed so far: $S_t = \operatorname{argmin}_{S\in\mathcal{S}} \sum_{\tau=1}^{t-1} f_\tau(S) + r(S)$ [19]. The last term is a regularizer (e.g., [1, 24]) or, in the simplest case, a random modular perturbation $r(S) = r \cdot \chi_S$ for a random vector r (Follow-the-perturbed-leader, FPL).

As mentioned above, the FPL framework does not in general combine with any approximation algorithm. We concentrate on a specific family of approximation algorithms: those that solve (1) for a tractable approximation $\hat{f}$ of f. We define two general conditions:

C1 The approximation $\hat{f}$ of f satisfies $f(A) \leq \hat{f}(A) \leq \alpha f(A)$ for all $A \subseteq E$.
C2 The following combinatorial problem can be solved exactly in polynomial time:

$$\operatorname*{argmin}_{S\in\mathcal{S}} \sum_t \hat{f}_t(S) + \alpha r(S). \tag{6}$$

Algorithm 2 integrates this $\hat{f}$ into the FPL framework: in each round, we play an expected minimizer of the approximate cost, $S_t \in \operatorname{argmin}_{S\in\mathcal{S}} \sum_{\tau=1}^{t-1} \hat{f}_t(S) + \alpha r(S)$. Condition (C2) ensures that we can find such an S_t. Contrary to the failure example in [18], the deviation between $\hat{f}_t(S)$ and $f_t(S)$ is bounded for each f_t, and not only for the sum. We get the following bound on the α-regret.

Theorem 2. *For an approximation $\hat{f}$ that satisfies (C1) and (C2), $M = \max_t f_t(E)$, and $\eta = T^{-1/2}$, Algorithm 2 achieves an expected α-regret $\mathbb{E}[R_\alpha(T)] \leq 3\alpha m M/\sqrt{T} = O(\alpha m/\sqrt{T})$.*

The conditions show that a suitable approximation $\hat{f}$ is decisive. For minimum spanning tree and perfect matching, the simple approximation $\hat{f}_m(S) = \sum_{e\in S} f(e)$ already leads to the best achievable approximation bound $O(|V|)$ [10]. In this case, the optimization problem (6) reduces to the standard version with a modular, sum-of-weights cost. For the multi-agent version in [10], we sum each agent's cost functions individually over time, and use this to greedily assign agents to edges.

The simple $\hat{f}_m$, however, leads in general to approximation factors of order $O(m)$. With more interesting approximations, we can possibly do better on other problems. Goemans et al. [11] propose a generic $\sqrt{m}\log m$ approximation for any non-decreasing submodular function. However, this approximation does not in general easily satisfy (C2), since it is usually used as $\hat{f}^2$.

For minimum (s,t)-cut, the structure of E and $\mathcal{C}$ comes to the rescue and leads to a different approximation [16]. The idea is to make f separable over certain structures to make it tractable, but to retain submodularity over restricted sets to improve on the approximation $\hat{f}_m$. Here, E is the set of directed edges in a graph $G = (V, E)$. We partition E by assigning an edge $e = (v, z)$ either to its tail v or head z. All possible such partitions form a family $\mathfrak{P}$ and are of the form $\Pi = \{E_v^\Pi\}_v$,

where the set $E_v^\Pi \subseteq E$ contains the edges assigned to node v. We retain submodularity within the sets E_v^Π, and make $\hat{f}$ separable across the E_v^Π:

$$\hat{f}(S) = \min_{\Pi \in \mathfrak{P}} \sum_{v \in V} f(S \cap E_v^\Pi). \tag{7}$$

This approximation corresponds to a convolution of submodular functions and can be minimized by solving the dual problem, a polymatroidal max-flow [20]. The approximation factor for (C1) is then bounded by the maximum number of nodes Δ_s on the s or Δ_t on the t side of the optimal cut that have an adjacent cut edge: $\alpha \leq \min\{|\Delta_s|, |\Delta_t|\} \leq |V|/2$ [16]; $f \leq \hat{f}$ holds by subadditivity of f.

A slight modification of the proposed $\hat{f}$ retains α in expectation and satisfies (C2). Instead of using the best partition in $\mathfrak{P}$, we fix Π. Let Π_h be the partition that assigns each edge to its head, and Π_t the one that assigns each edge to its tail. The minimum cut for Π_h, $\hat{f}_h(S) = \sum_{v \in V} f(S \cap E_v^{\Pi_h})$, can still be solved as a polymatroidal flow, and equivalently for Π_t. A small modification of the proof in [16] shows that the approximation factor for $\hat{f}_h$ and $\hat{f}_t$ is $|\Delta_t|$ and $|\Delta_s|$, respectively. Before starting the algorithm, we randomly pick either $\hat{f}_h$ or $\hat{f}_t$ and use it throughout. Then $\mathbb{E}[\alpha] = \mathbb{E}[f(S')/f(S)] \leq (|\Delta_t| + |\Delta_s|)/2 \leq |V|/2$ for (C1). With a fixed partition, it holds that $\sum_t \hat{f}_t + r = \widehat{(\sum_t f_t)} + r$, and the latter can be minimized exactly by a polymatroidal flow. Thus, (C2) is satisfied.

Proof. (Thm. 2, outline) The proof builds on [12, 19] and uses (C1) and (C2). Let

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

We adapt the proof of Lemma 3.1 in [19] and obtain

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

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

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

Condition (C1) implies that

$$\sum\nolimits_{t=1}^{T} \hat{f}_t(\widehat{S}_{T+1}) \leq \sum\nolimits_{t=1}^{T} \hat{f}_t(S_T^*) \leq \alpha \sum\nolimits_{t=1}^{T} f_t(S_T^*), \tag{8}$$

and, for the left hand side, that $\sum_{t=1}^{T} f_t(S_t) \leq \sum_{t=1}^{T} \hat{f}_t(S_t)$. This yields the regret bound

$$\sum\nolimits_{t=1}^{T} f_t(S_t) - \alpha \sum_{t=1}^{T} f_t(S_T^*) \leq \sum\nolimits_{t=1}^{T} (\hat{f}_t(S_t) - \hat{f}_t(S_{t+1})) + \alpha(r(\widehat{S}_{T+1}) - r(S_1)).$$

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 adapt a technique in [12] to the approximate setting (pruned for space reasons) to bound the probability that $S_t \neq S_{t+1}$, and reach at the inequality $\sum_{t=1}^{T} \mathbb{E}[\hat{f}_t(S_t) - \hat{f}_t(S_{t+1})] \leq \sum_{t=1}^{T} P(S_t \neq S_{t+1}) \max_{A \in \mathcal{S}} \hat{f}(A) \leq 2\alpha m M T \eta$. Finally, combining these bounds yields

$$\mathbb{E}\big[\sum\nolimits_{t=1}^{T} f_t(S_t)\big] - \alpha \sum\nolimits_{t=1}^{T} f_t(S_T^*) \leq \alpha M m/\eta + 2\alpha m M T \eta. \tag{9}$$

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

4 Conclusion

We outlined two routes for deriving online approximation algorithms for submodular minimization with combinatorial constraints. First, convex relaxations with rounding are amenable to a subgradient descent with rounding. Second, for approximations of the cost function, we show two conditions to use a Follow-the-leader algorithm. Depending on the problem at hand, one or the other approach may be more suitable. Further work will address better and even more general algorithms. Moreover, beyond the full information case, bandit algorithms for the setting discussed here face the challenges of nonlinearity and nonseparability together with combinatorial constraints.

References

[1] J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit optimization. In *COLT*, 2008.

[2] B. Awerbuch and R. D. Kleinberg. Adaptive routing with end-to-end feedback: distributed learning and geometric approaches. In *STOC*, 2004.

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

[4] N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. In *COLT*, 2009.

[5] F. A. Chudak and K. Nagano. Efficient solutions to relaxations of combinatorial problems with submodular penalties via the lovász extension and non-smooth convex optimization. In *SODA*, 2007.

[6] V. Dani, T. P. Hayes, and S. M. Kakade. The price of bandit information for online optimization. In *NIPS*, 2008.

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

[8] L. Fleischer. Recent progress in submodular function minimization. *Optima: Mathematical Programming Society Newsletter*, (64), September 2000.

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

[10] G. Goel, C. Karande, P. Tripati, and L. Wang. Approximability of combinatorial problems with multiagent submodular cost functions. In *FOCS*, 2009.

[11] M. X. Goemans, N. J. A. Harvey, A. Iwata, and V. S. Mirrokni. Approximating submodular functions everywhere. In *SODA*, 2009.

[12] E. Hazan and S. Kale. Online submodular minimization. In *NIPS*, 2009.

[13] D. P. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. *JMLR*, 10:1705–1736, 2009.

[14] D. Hochbaum. Submodular problems – approximations and algorithms. *arXiv:1010.1945.v1*, 2010.

[15] S. Iwata and K. Nagano. Submodular function minimization under covering constraints. In *FOCS*, 2009.

[16] S. Jegelka and J. Bilmes. Cooperative cuts: graph cuts with submodular edge weights. Technical Report TR-189, Max Planck Institute for Biological Cybernetics, 2010.

[17] S. Jegelka and J. Bilmes. Cooperative cuts for image segmentation. Technical Report 2010-0003, University of Washington, 2010.

[18] S. Kakade, A. T. Kalai, and K. Ligett. Playing games with approximation algorithms. *SIAM J. Comput.*, 39(3):1088–1106, 2009.

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

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

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

[22] G. Lugosi, O. Papaspiliopoulos, and G. Stoltz. Online multi-task learning with hard constraints. In *COLT*, 2009.

[23] C. Seshadri and J. Vondrák. Is submodularity testable? In *arXiv 1008.0831v1*, 2010.

[24] S. Shalev-Shwartz and Y. Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 69(2-3):773–818, 2007.

[25] M. Streeter and D. Golovin. An online algorithm for maximizing submodular functions. In *NIPS*, 2008.

[26] M. Streeter, D. Golovin, and A. Krause. Online learning of assignments. In *NIPS*, 2009.

[27] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *FOCS*, 2008.

[28] M. Zinkevich. Online convex programming and infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.