Submodular Function Maximization

Andreas Krause (ETH Zurich)

Daniel Golovin (Google)

Submodularity[1] is a property of set functions with deep theoretical consequences and far–reaching applications. At first glance it appears very similar to concavity, in other ways it resembles convexity. It appears in a wide variety of applications: in Computer Science it has recently been identified and utilized in domains such as viral marketing (Kempe et al., 2003), information gathering (Krause and Guestrin, 2007), image segmentation (Boykov and Jolly, 2001; Kohli et al., 2009; Jegelka and Bilmes, 2011a), document summarization (Lin and Bilmes, 2011), and speeding up satisfiability solvers (Streeter and Golovin, 2008). In this survey we will introduce submodularity and some of its generalizations, illustrate how it arises in various applications, and discuss algorithms for optimizing submodular functions. Our emphasis here is on maximization; there are many important results and applications related to minimizing submodular functions that we do not cover[2].

As a concrete running example, we will consider the problem of deploying sensors in a drinking water distribution network (see Figure 1) in order to detect contamination. In this domain, we may have a model of how contaminants, accidentally or maliciously introduced into the network, spread over time. Such a model then allows to quantify the benefit $f(A)$ of deploying sensors at a particular set A of locations (junctions or pipes in the network) in terms of the detection performance (such as average time to detection). Based on this notion of utility, we then wish to find an optimal subset $A \subseteq V$ of locations maximizing the utility, $\max_A f(A)$, subject to some constraints (such as bounded cost). This application requires solving a difficult real-world optimization problem, that can be handled with the techniques discussed in this chapter (Krause et al. 2008b show in detail how submodular optimization can be applied in this domain.) We will also discuss more complex settings, for example how one can incorporate complex constraints on the feasible sets A, robustly optimize against adversarially chosen objective functions f, or adaptively select sensors based on previous observations.

Several algorithms for submodular optimization described in this survey are implemented in an open source Matlab toolbox[3] (Krause, 2010).

1 Submodular functions

Submodularity is a property of *set functions*, i.e., functions $f : 2^V \to \mathbb{R}$ that assign each subset $S \subseteq V$ a value $f(S)$. Hereby V is a finite set, commonly called the *ground set*. In our example, V may refer to the locations where sensors can be placed, and $f(S)$ the utility (e.g., detection performance) obtained when placing sensors at locations S. In the following, we will also assume that $f(\emptyset) = 0$, i.e., the empty set carries no value. Submodularity has two equivalent definitions, which we will now describe. The first definition relies on a notion of discrete derivative, often also called the marginal gain.

Definition 1.1 (Discrete derivative) For a set function $f : 2^V \to \mathbb{R}$, $S \subseteq V$, and $e \in V$, let $\Delta_f(e \mid S) := f(S \cup \{e\}) - f(S)$ be the *discrete derivative* of f at S with respect to e.

Where the function f is clear from the context, we drop the subscript and simply write $\Delta(e \mid S)$.

Definition 1.2 (Submodularity) A function $f : 2^V \to \mathbb{R}$ is *submodular* if for every $A \subseteq B \subseteq V$ and $e \in V \setminus B$ it holds that

$$\Delta(e \mid A) \geq \Delta(e \mid B)\,.$$

Equivalently, a function $f : 2^V \to \mathbb{R}$ is *submodular* if for every $A, B \subseteq V$,

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B).$$

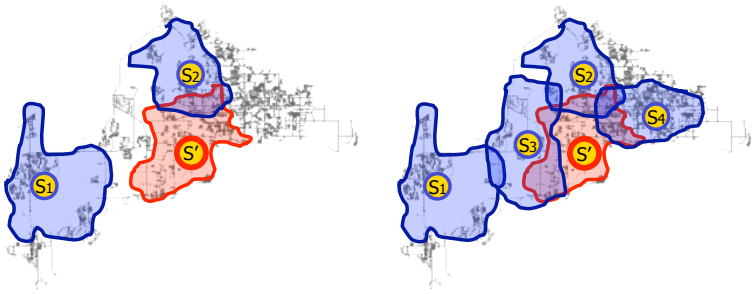
For submodular maximization, the intuition provided by the first definition is often helpful: Suppose we interpret $S \subset V$ as a set of actions which provide some benefit $f(S)$. Then the first definition says that for a submodular function f, after performing a set A of actions, the marginal benefit of any action e does not increase as we perform the actions in $B \setminus A$. Therefore, submodular set functions exhibit a natural diminishing returns property. Figure 1 illustrates this effect in our sensor placement application. In this example, the marginal benefit provided by placing a sensor at a fixed location s' given that we deployed sensors at locations s_1, s_2 does not increase as we deploy more sensors (s_3 and s_4).

An important subclass of submodular functions are those which are *monotone*, where enlarging the argument set cannot cause the function to decrease.

Definition 1.3 (Monotonicity) A function $f : 2^V \to \mathbb{R}$ is *monotone* if for every $A \subseteq B \subseteq V$, $f(A) \leq f(B)$.

Note that a function f is monotone iff all its discrete derivatives are nonnegative, i.e., iff for every $A \subseteq V$ and $e \in V$ it holds that $\Delta(e \mid A) \geq 0$. Further note that the important subclass of monotone submodular functions can be characterized by requiring that for all $A \subseteq B \subseteq V$ and $e \in V$ it holds that $\Delta(e \mid A) \geq \Delta(e \mid B)$. This is slightly different from Definition 1.2 in that we do not require $e \notin B$.

Typically, and in most of this chapter, we will assume that f is given in terms of a *value oracle*, a black box that computes[4] $f(S)$ on any input set S.



(a) *Adding s' to set $\{s_1, s_2\}$* (b) *Adding s' to superset $\{s_1, \ldots, s_4\}$*

Figure 1 Illustration of the diminishing returns effect in context of placing sensors in a water distribution network to detect contaminations. The blue regions indicate nodes where contamination is detected quickly using the existing sensors S. The red region indicates the additional coverage by adding a new sensor s'. If more sensors are already placed (b), there is more overlap, hence less gain in utility: $\Delta(s' \mid \{s_1, s_2\}) \geq \Delta(s' \mid \{s_1, \ldots, s_4\})$.

1.1 Examples

Submodular functions comprise a broad class of functions that arise in several applications. Here are some examples.

Modular functions and generalizations. The simplest example of submodular functions are *modular* functions, those for which the inequalities characterizing submodularity hold with equality, i.e., for all $A, B \subseteq V$ it holds that $f(A) + f(B) = f(A \cup B) + f(A \cap B)$. Such functions are analogous to linear functions, insofar as their discrete derivatives are constant: $\Delta(e \mid B) = \Delta(e \mid A)$ for all A, B and $e \notin A \cup B$. Assuming $f(\emptyset) = 0$, they can always be expressed in the form $f(S) = \sum_{e \in S} w(e)$ for some weight function $w : V \to \mathbb{R}$. Another example is the composition of any monotone modular function $g : 2^V \to \mathbb{R}$ and any concave function $h : \mathbb{R} \to \mathbb{R}$ — for example, $f(S) = \sqrt{|S|}$.

Weighted coverage functions. An important example of a submodular function is the *weighted coverage* of a collection of sets: Fix a set X, a nonnegative modular function $g : 2^X \to \mathbb{R}$, and a collection V of subsets of X. Then for a subcollection $S \subseteq V$, the function

$$f(S) := g\Big(\bigcup_{v \in S} v\Big) = \sum_{x \in \bigcup_{v \in S} v} w(x),$$

is monotone submodular. Hereby $w : X \to \mathbb{R}$ is the weight function representing g. In our example, X may refer to a set of contamination events, $w(x)$ quantifies the severity of event x, and with each possible sensor location $v \in V$ we associate the subset $v \subseteq X$ of events detected. Perhaps the simplest example is where $g(A) = |A|$ is the cardinality function (which is modular), in which case the problem of maximizing $f(S)$ is the well-known max-cover problem. In fact, $f(S)$ is submodular even for arbitrary *submodular* functions g. It is monotone iff g is monotone.

The rank function of a matroid. Another important class of submodular functions arises in the context of matroids:

Definition 1.4 (Matroid) A *matroid* is a pair $(V, \mathcal{I})$ such that V is a finite set, and $\mathcal{I} \subseteq 2^V$ is a collection of subsets of V satisfying the following two properties:

- $A \subseteq B \subseteq V$ and $B \in \mathcal{I}$ implies $A \in \mathcal{I}$
- $A, B \in \mathcal{I}$ and $|B| > |A|$ implies $\exists\, e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

Sets in $\mathcal{I}$ are called *independent*, and matroids generalize the concept of linear independence found in linear algebra. An important function associated with a matroid $(V, \mathcal{I})$, which describes it completely, is its *rank function* $f(S) := \max\{|U| : U \subseteq S, U \in \mathcal{I}\}$. The rank function of any matroid is monotone submodular (Birkhoff, 1933).

Facility location. Suppose we wish to select, out of a set $V = \{1, \ldots, n\}$, some locations to open up facilities in order to serve a collection of m customers. If we open up a facility at location j, then it provides service of value $M_{i,j}$ to customer i, where $M \in \mathbb{R}^{m \times n}$. If each customer chooses the facility with highest value, the total value provided to all customers is modeled by the set function

$$f(S) = \sum_{i=1}^{m} \max_{j \in S} M_{i,j}.$$

Hereby we set $f(\emptyset) = 0$. If $M_{i,j} \geq 0$ for all i, j, then $f(S)$ is monotone submodular (Frieze, 1974). This model is quite general, and captures other applications as well. In our sensor placement example, $M_{i,j}$ could refer to the benefit provided by sensor j in scenario i, quantified, e.g., in terms of the expected reduction in detection time (Krause et al., 2008b).

Entropy. Given a joint probability distribution $P(\mathbf{X})$ over a discrete-valued random vector $\mathbf{X} = [X_1, X_2, \ldots, X_n]$, the function $f(S) = H(\mathbf{X}_S)$ is monotone submodular (Fujishige, 1978), where H is the Shannon entropy, i.e.,

$$H(\mathbf{X}_S) = -\sum_{\mathbf{x}_S} P(\mathbf{x}_S) \log_2 P(\mathbf{x}_S)$$

where we use the notational convention that $\mathbf{X}_S$ is the random vector consisting of the coordinates of $\mathbf{X}$ indexed by S, and likewise $\mathbf{x}_S$ is the vector consisting of the coordinates of an assignment $\mathbf{x}$ indexed by S. If the random variables are real-valued, with a probability density function f, the differential entropy

$$H(\mathbf{X}_S) = -\int P(\mathbf{x}_S) \log_2 P(\mathbf{x}_S) d\mathbf{x}_S$$

is submodular as well, but not generally monotone.

Mutual information. Given a joint probability distribution $P(\mathbf{X}, \mathbf{Y})$ over two dependent random vectors, $\mathbf{X} = [X_1, X_2, \ldots, X_n]$ and $\mathbf{Y} = [Y_1, Y_2, \ldots, Y_m]$, consider the *mutual information*: $f(S) = I(\mathbf{Y}; \mathbf{X}_S) = H(\mathbf{Y}) - H(\mathbf{Y} \mid \mathbf{X}_S)$, which quantifies the expected reduction of uncertainty about $\mathbf{Y}$ upon revelation of $\mathbf{X}_S$. In general, the function f is *not submodular*: Suppose $X_1, X_2 \sim$ Bernoulli(0.5), and $Y = X_1$ XOR X_2. Then $f(\emptyset) = f(\{1\}) = f(\{2\}) = 0$, but $f(\{1, 2\}) = 1$, violating submodularity. However, if the variables $\mathbf{X}$ are *conditionally independent* given $\mathbf{Y}$, i.e., for all disjoint sets $A, B \subset V$ if holds that $\mathbf{X}_A \perp \mathbf{X}_B \mid \mathbf{Y}$, then f is monotone submodular (Krause and Guestrin, 2005). This holds both for discrete and continuous distributions. In our example, we may associate a variable Y_v with the water quality at location $v \in V$, and X_v is a noisy measurement of Y_v that we obtain if we place a sensor at location v. Then $f(S)$ quantifies how much we can reduce our uncertainty about the water quality everywhere when deploying sensors at locations S.

Symmetric mutual information. Let $V = \{1, 2, \ldots, n\}$. Given any joint probability distribution P over a random vector $\mathbf{X} = [X_1, X_2, \ldots, X_n]$, the function $f(S) = I(\mathbf{X}_S; \mathbf{X}_{V \setminus S})$ is submodular. However, f is not monotone in general, since $f(\emptyset) = f(V) = 0$, and unless $X_1, X_2, \ldots, X_n$ are independent, it will be the case that $f(S) > 0$ for some S. This function has been used by Narasimhan et al. (2005) for information-theoretic clustering problems, and by Krause et al. (2008a) for the purpose of sensor placement.

More generally, if $\mathbf{Y} = [Y_1, Y_2, \ldots, Y_m]$ is another random vector, and $\mathbf{X}$ and $\mathbf{Y}$ have joint distribution $P(\mathbf{X}, \mathbf{Y})$ the *conditional mutual information* $I(\mathbf{X}_S; \mathbf{X}_{V \setminus S} \mid \mathbf{Y})$ is submodular (but not monotone). This function arises in the context of structure learning in probabilistic graphical models, as studied by Narasimhan and Bilmes (2004).

Cut capacity functions. Fix any undirected graph $G = (V, E)$ with nonnegative edge capacities $c : E \to \mathbb{R}_+$. Let ∂S be the *boundary* of $S \subseteq V$, defined as $\partial S := \{\{u, v\} \in E : |S \cap \{u, v\}| = 1\}$. Then the function $f(S) = \sum_{e \in \partial S} c(e)$ is submodular (Schrijver, 2003). The same is true for directed graphs, if we define $\partial S := \{(u, v) \in E : u \in S, v \notin S\}$. Note that f is not generally monotone: In particular, $f(\emptyset) = f(V) = 0$.

1.2 Properties of submodular functions

Submodular functions have many useful properties. For example, submodularity is preserved under taking nonnegative linear combinations. In other words, if $g_1, \ldots, g_n : 2^V \to \mathbb{R}$ are submodular, and $\alpha_1, \ldots, \alpha_n \geq 0$, then $f(S) := \sum_{i=1}^{n} \alpha_i g_i(S)$ is submodular as well. This is readily proved using the definition of submodularity based on discrete derivatives. This insight is extremely useful, as it allows to build complex submodular objectives from simpler constituents (*c.f.*, Kempe et al. 2003; Leskovec et al. 2007; Stobbe and Krause 2010). Submodularity is also preserved when we take the *residual*: if $g : 2^V \to \mathbb{R}$ is submodular, and $A, B \subset V$ are any disjoint sets, then the residual $f : 2^A \to \mathbb{R}$ defined via $f(S) := g(S \cup B) - g(B)$ is submodular. Monotone submodular functions remain so under *truncation*: if $g : 2^V \to \mathbb{R}$ is submodular, so is $f(S) := \min\{g(S), c\}$ for any constant c. While truncation preserves submodularity, in general, the minimum and maximum of two submodular functions are not submodular, i.e., for submodular functions f_1 and f_2, the

functions $f_{\min}(S) = \min(f_1(S), f_2(S))$ and $f_{\max}(S) = \max(f_1(S), f_2(S))$ are not necessarily submodular.

Interestingly, there are many natural connections between submodular functions and both convex and concave functions. For example, for a function $g : \mathbb{N} \to \mathbb{R}$, the set function $f(S) = g(|S|)$ is submodular if and only if g is concave. In contrast, similar to convex functions, which can be minimized efficiently, (unconstrained) submodular minimization is possible in (strongly) polynomial time (*c.f.*, Schrijver 2003). See (Lovasz, 1983) for a discussion about the relationship between submodular, concave and convex functions.

Submodular set functions can also be extended to continuous functions (defined over the unit cube $[0, 1]^{|V|}$) in several natural ways. See Section 3.2 for more details on such extensions.

2 Greedy maximization of submodular functions

As argued in Section 1.1, submodular functions arise in many applications, and therefore it is natural to study submodular optimization. There is a large amount of work on minimizing submodular functions (*c.f.*, Fujishige 2005; Schrijver 2003). In this chapter, we will focus on the problem of maximizing submodular functions. That is, we are interested in solving problems of the form

$$\max_{S \subseteq V} f(S) \text{ subject to some constraints on } S. \tag{1}$$

The simplest example are *cardinality constraints*, where we require that $|S| \leq k$ for some k. In our example, we may wish to identify the k best locations to place sensors. Unfortunately, even this simple problem is NP-hard, for many classes of submodular functions, such as weighted coverage (Feige, 1998) or mutual information (Krause and Guestrin, 2005). While there are specialized branch and bound algorithms for maximizing submodular functions (Nemhauser and Wolsey, 1981; Goldengorin et al., 1999; Kawahara et al., 2009), ultimately their scalability is limited by the hardness of Problem 1. Therefore, in the remaining of this chapter we focus on efficient algorithms with theoretical approximation guarantees.

The greedy algorithm. In the following, we will consider the problem of approximately maximizing monotone submodular functions. A simple approach towards solving Problem 1 in the case of cardinality constraints is the *greedy algorithm*, which starts with the empty set S_0, and in iteration i, adds the element maximizing the discrete derivative $\Delta(e \mid S_{i-1})$ (ties broken arbitrarily):

$$S_i = S_{i-1} \cup \{\arg\max_e \Delta(e \mid S_{i-1})\}. \tag{2}$$

A celebrated result by Nemhauser et al. (1978) proves that the greedy algorithm provides a good approximation to the optimal solution of the NP-hard optimization problem.

Theorem 1.5 (Nemhauser et al. 1978) *Fix a nonnegative monotone submodular function $f : 2^V \to \mathbb{R}_+$ and let $\{S_i\}_{i \geq 0}$ be the greedily selected sets defined in Eq. (2). Then for all*

positive integers k and ℓ,

$$f(S_\ell) \geq \left(1 - e^{-\ell/k}\right) \max_{S:|S|\leq k} f(S).$$

In particular, for $\ell = k$, $f(S_k) \geq (1 - 1/e) \max_{|S|\leq k} f(S)$.

Proof Nemhauser et al. only discussed the case $\ell = k$, however their very elegant argument easily yields the slight generalization above. It goes as follows. Fix ℓ and k. Let $S^* \in \arg\max \{f(S) : |S| \leq k\}$ be an optimal set of size k (due to monotonicity of f we can assume w.l.o.g. it is of size exactly k), and order the elements of S^* arbitrarily as $\{v_1^*, \ldots, v_k^*\}$. Then we have the following sequence of inequalities for all $i < \ell$, which we explain below.

$$\begin{aligned}
f(S^*) &\leq f(S^* \cup S_i) && (3)\\
&= f(S_i) + \sum_{j=1}^{k} \Delta\left(v_j^* \mid S_i \cup \left\{v_1^*, \ldots, v_{j-1}^*\right\}\right) && (4)\\
&\leq f(S_i) + \sum_{v \in S^*} \Delta(v \mid S_i) && (5)\\
&\leq f(S_i) + \sum_{v \in S^*} \left(f(S_{i+1}) - f(S_i)\right) && (6)\\
&\leq f(S_i) + k\left(f(S_{i+1}) - f(S_i)\right) && (7)
\end{aligned}$$

Eq. (3) follows from monotonicity of f, Eq. (4) is a straightforward telescoping sum, Eq. (5) follows from the submodularity of f, Eq. (6) holds because S_{i+1} is built greedily from S_i in order to maximize the marginal benefit $\Delta(v \mid S_i)$, and Eq. (7) merely reflects the fact that $|S^*| \leq k$. Hence

$$f(S^*) - f(S_i) \leq k\left(f(S_{i+1}) - f(S_i)\right). \quad (8)$$

Now define $\delta_i := f(S^*) - f(S_i)$, which allows us to rewrite Eq. (8) as $\delta_i \leq k\left(\delta_i - \delta_{i+1}\right)$, which can be rearranged to yield

$$\delta_{i+1} \leq \left(1 - \frac{1}{k}\right) \delta_i \quad (9)$$

Hence $\delta_\ell \leq \left(1 - \frac{1}{k}\right)^\ell \delta_0$. Next note that $\delta_0 = f(S^*) - f(\emptyset) \leq f(S^*)$ since f is nonnegative by assumption, and by the well-known inequality $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$ we have

$$\delta_\ell \;\leq\; \left(1 - \frac{1}{k}\right)^\ell \delta_0 \;\leq\; e^{-\ell/k} f(S^*). \quad (10)$$

Substituting $\delta_\ell = f(S^*) - f(S_\ell)$ and rearranging then yields the claimed bound of $f(S_\ell) \geq \left(1 - e^{-\ell/k}\right) f(S^*)$. □

The slight generalization allowing $\ell \neq k$ is quite useful. For example, if we let the greedy algorithm pick $5k$ sensors, the approximation ratio (compared to the optimal set of size k) improves from $\approx .63$ to $\approx .99$.

For several classes of submodular functions, this result is the best that can be achieved with any efficient algorithm. In fact, Nemhauser and Wolsey (1978) proved that any algorithm that is allowed to only evaluate f at a polynomial number of sets will not be

able to obtain an approximation guarantee better than $(1 - 1/e)$. (Subsequently, Vondrák (2010) provided more refined results on the best possible approximation factor in terms of a parameter called the *curvature* of f.)

Matroid constraints. Going beyond cardinality constraints, the greedy algorithm is also guaranteed to provide near-optimal solutions for more complex constraints. In our sensing example, in order to preserve energy, each sensor may decide when to activate. The problem of optimizing such a sensing schedule requires maximizing a submodular function subject to a partition matroid constraint (Krause et al., 2009).

Suppose $(V, \mathcal{I})$ is a matroid, and we wish to solve the problem

$$\max_{S \in \mathcal{I}} f(S),$$

then the greedy algorithm, which starts with S_G and sets

$$S_G \leftarrow S_G \cup \left\{ \operatorname*{arg\,max}_{e \notin S_G : S_G \cup \{e\} \in \mathcal{I}} \Delta(e \mid S_G) \right\} \tag{11}$$

until there is no more e such that $S_G \cup \{e\} \in \mathcal{I}$ (i.e., there is no element which can be added to create a feasible solution), is guaranteed to produce a solution S_G so that $f(S_G) \geq \frac{1}{2} \max_{S \in \mathcal{I}} f(S)$.

Even more generally, suppose $(V, \mathcal{I}_1), \ldots, (V, \mathcal{I}_p)$ are p matroids, and $\mathcal{I} = \bigcap_i \mathcal{I}_i$. That is, $\mathcal{I}$ consists of all subsets of V that are independent in all p matroids. Even though $(V, \mathcal{I})$ is not generally a matroid anymore, the greedy algorithm 11 is guaranteed to produce a solution so that $f(S_G) \geq \frac{1}{p+1} \max_{S \in \mathcal{I}} f(S)$. In fact, this results holds even more generally whenever $(V, \mathcal{I})$ is a *p-extensible system*, a combinatorial notion which generalizes the intersections of p matroids (Calinescu et al., 2011).

Min-cost coverage. Instead of maximizing a monotone submodular function subject to constraints, it is also natural to search for minimum cost sets that achieve a given amount q of submodular value. In particular, we may wish to solve

$$S^* = \operatorname*{arg\,min}_S |S| \text{ s.t. } f(S) \geq q, \tag{12}$$

for some quota $0 \leq q \leq f(V)$ of value. In our sensing example, we may wish to deploy as few sensors as possible, while guaranteeing that all possible contamination scenarios are eventually detected.

Wolsey (1982) proves the following result about the greedy algorithm:

Theorem 1.6 (Wolsey 1982) *Suppose $f : 2^V \to \mathbb{N}$ is monotone submodular and integer-valued, and let $0 \leq q \leq f(V)$. Let $S_0, S_1, \ldots$ be the sequence of sets picked by the greedy algorithm, and let ℓ be the smallest index such that $f(S_\ell) \geq q$. Then*

$$\ell \leq \left(1 + \ln \max_{v \in V} f(\{v\})\right) OPT,$$

where $OPT = \min_S |S|$ s.t. $f(S) \geq q$.

In fact, Wolsey proves the special case $q = f(V)$, but the result above immediately follows by applying his result to the submodular function $\min\{f(S), q\}$. Wolsey also proves that the same result holds in the case where the elements of V have non-uniform cost, using a slightly modified greedy algorithm (see also Section 3.1).

Speeding up the greedy algorithm through lazy evaluations. In some applications, evaluating the function f can be expensive. In our example, evaluating f may require running computationally costly water quality simulations. In this case, even applying the standard greedy algorithm can be infeasible. Fortunately, submodularity can be exploited algorithmically to implement an accelerated variant of the greedy algorithm, originally proposed by Minoux (1978). In each iteration i, the greedy algorithm must identify the element e with maximum marginal gain $\Delta(e \mid S_{i-1})$, where S_{i-1} is the set of elements selected in the previous iterations. The key insight is that, as a consequence of submodularity of f, the marginal benefits of any fixed element $e \in V$ are monotonically nonincreasing during the iterations of the algorithm, i.e., $\Delta(e \mid S_i) \geq \Delta(e \mid S_j)$ whenever $i \leq j$. Instead of recomputing $\Delta(e \mid S_{i-1})$ for each element $e \in V$ (requiring $O(n)$ computations of f), the accelerated greedy algorithm maintains a list of upper bounds $\rho(e)$ (initialized to ∞) on the marginal gains sorted in decreasing order. In each iteration, the algorithm extracts the maximal element

$$e \in \operatorname*{arg\,max}_{e' : S_{i-1} \cup \{e'\} \in \mathcal{I}} \rho(e')$$

from the ordered list. It then updates the bound $\rho(e) \leftarrow \Delta(e \mid S_{i-1})$. If, after this update, $\rho(e) \geq \rho(e')$ for all $e' \neq e$, then submodularity guarantees that $\Delta(e \mid S_{i-1}) \geq \Delta(e' \mid S_{i-1})$ for all $e' \neq e$, and therefore the greedy algorithm has identified the element of largest marginal gain, without having to compute $\Delta(e' \mid S_{i-1})$ for a potentially large number of elements e'. It sets $S_i \leftarrow S_{i-1} \cup \{e\}$ and repeats until there is no further feasible element which can be added. This idea of using lazy evaluations can lead to orders of magnitude performance speedups, and is useful beyond the greedy algorithm (*c.f.*, Leskovec et al. 2007).

3 Beyond the greedy algorithm: Handling more complex constraints

In this section, we will survey some work on submodular optimization beyond the standard greedy algorithm discussed in Section 2. Using more complex algorithms allows to handle maximization subject to more complex constraints. We will mostly focus on the case of *monotone* functions, but also mention results about optimizing non-monotone functions.

3.1 Knapsack constraints

Instead of selecting a set of at most k elements, in many applications, the elements $v \in V$ may have non-uniform costs $c(s) \geq 0$, and we may wish to maximize f subject to a budget that the total cost cannot exceed:

$$\max_S f(S) \text{ s.t. } \sum_{v \in S} c(v) \leq B.$$

Thus, we would like to maximize $f(S)$ subject to a (w.l.o.g.) nonnegative modular constraint, also called Knapsack constraint (as the problem of maximizing a *modular* f subject to a *modular* constraint c is called the Knapsack problem). Naturally, the standard (uniform cost) greedy algorithm, selecting the next affordable element of maximum marginal gain can perform arbitarily badly, as it ignores cost. It can be easily modified to take cost into account: The cost-benefit greedy algorithm starts with $S_0 = \emptyset$, and iteratively adds

$$S_{i+1} = S_i \cup \left\{ \operatorname*{arg\,max}_{v \in V \setminus S_i : c(v) \leq B - c(S_i)} \frac{\Delta(e \mid S_i)}{c(v)} \right\}, \tag{13}$$

i.e., the element v that maximizes the benefit cost ratio among all elements still affordable with the remaining budget. For the min-cost covering problem (12), Wolsey (1982) proves a generalization of Theorem 1.6 for this cost-benefit greedy algorithm. Unfortunately, for the budgeted maximization problem, even though this modified algorithm takes cost into account, it can still perform arbitrarily badly. However, perhaps surprisingly, *at least one* of the greedy solutions – the solution S_{uc} returned by the uniform cost or the one S_{cb} provided by the cost-benefit greedy algorithm cannot perform too badly: it can be shown that $\max\{f(S_{uc}), f(S_{cb})\} \geq \frac{1-1/e}{2} OPT$ (Leskovec et al., 2007). In fact, a more computationally complex algorithm, which enumerates all sets S of size 3, and augments them using the cost-benefit greedy algorithm, is known to provide a $1 - 1/e$ approximation (Sviridenko, 2004).

3.2 Submodular maximization using the multilinear extension

One important idea, which has seen a number of applications in submodular optimization, is the use of *extensions*. Suppose $f : 2^V \to \mathbb{R}$ is a set function. By identifying sets S with binary vectors $\mathbf{e}_S$ (in which the i-th component is 1 if $i \in S$, and 0 otherwise), we can equivalently represent f as a function defined over corners of the unit cube: $\tilde{f} : \{0,1\}^n \to \mathbb{R}$, where $n = |V|$, and $\tilde{f}(\mathbf{e}_S) = f(S)$. From this perspective, it is natural to extend $\tilde{f}$ to the entire unit cube $[0,1]^n$. There are several important extensions. The *Lovaśz extension* (Lovasz, 1983) extends $\tilde{f}$ to a *convex* function $\hat{f} : [0,1]^n \to \mathbb{R}$, for which (at least some of) its minimizers are attained at a corner of $[0,1]^n$. Minimization of $\hat{f}$ over $[0,1]^n$ is possible using the ellipsoid method, which proved that unconstrained submodular minimization is possible in polynomial time. However, for the purpose of (constrained) submodular *maximization*, a different extension, pioneered by Vondrák (2008) has proven to be very useful: The *multilinear extension* of f, $\hat{f} : [0,1]^n \to \mathbb{R}$ is defined as

$$\hat{f}(\mathbf{x}) = \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{j \notin S} (1 - x_j).$$

Thus, $\hat{f}(\mathbf{x})$ is the expected value of f over sets, where each element i is included independently with probability x_i. Several recent algorithms for submodular maximization are built around (approximately) solving the problem

$$\max \hat{f}(\mathbf{x}) \text{ s.t. } \mathbf{x} \in \mathcal{F}$$

over some domain $\mathcal{F} \subseteq [0,1]^n$, and then rounding the continuous solution to obtain a near-optimal set.

The first application of this elegant idea, due to Vondrák (2008), is the *continuous greedy algorithm* for maximizing a submodular function subject to matroid constraints. In this application, the feasible set $\mathcal{F}$ is the *matroid polytope*, the convex hull of the independent sets of the underlying matroid. Conceptually, the algorithm traces the continuous particle, parameterized as a function $\mathbf{x} : [0,1] \rightarrow \mathcal{F}$, originating at $\mathbf{x}(0) = \mathbf{0}$, and following the differential equation

$$\dot{\mathbf{x}} = \arg\max_{\mathbf{v} \in \mathcal{F}} \left(\mathbf{v}^T \cdot \nabla \hat{f}(\mathbf{x}) \right).$$

Calinescu et al. (2011) prove that for matroid polytopes $\mathcal{F}$, it holds that

$$\hat{f}(\mathbf{x}(1)) \geq (1 - 1/e) \max_{\mathbf{x} \in \mathcal{F}} \hat{f}(\mathbf{x}),$$

thus at time 1, the particle has reached a point which provides a $(1 - 1/e)$ approximation of the optimal value of $\hat{f}$ over $\mathcal{F}$. Calinescu et al. also show how this continuous differential equation can be approximated by a discrete process up to arbitrarily small error. They also show how the continuous solution $\mathbf{y}$ can be efficiently rounded to a feasible discrete solution without loss in objective value, using *pipage rounding* (Ageev and Sviridenko, 2004). This result affirmatively closed the long-standing open question of whether the optimal approximation ratio of $(1 - 1/e)$, which the standard greedy algorithm achieves for cardinality constraints, can be achieved for *arbitrary* matroids (for which the standard algorithm only gives a $1/2$ approximation).

However, this general technique of relaxing constrained submodular maximization to a continuous problem, and rounding the obtained solution has proven to be far more general. For example, Kulik et al. (2009) have shown how to obtain a $(1-1/e-\varepsilon)$ approximation for the problem of maximizing a monotone submodular functions subject to multiple knapsack constraints. Recently, Chekuri et al. (2011) have used the multilinear relaxation to obtain a $.38/k$ approximation for maximizing a monotone submodular function subject to k matroid and a constant number of knapsack constraints (as well as an even more general class of other *downward-closed* constraints).

3.3 Submodular optimization over graphs

Another natural class of constraints arise when solving submodular optimization problems on graphs. Suppose that we identify the elements V as vertices of a (weighted) graph $G = (V, E, w)$ with edges E, and a function w that assigns each edge a nonnegative weight. In this setting, we may wish to maximize a submodular function $f(S)$ defined over the vertices, subject to the constraint that the set S forms a path, or a tree on G of weight at most B. Similarly, we may wish to obtain a tree (path) on G of submodular value q, and with approximately minimal total weight. These problems have natural applications in placing sensors under communication constraints, where the vertices V denote possible locations for sensors, f the informativeness of having sensors at a set of locations, and edges and their weights denote the communication cost between arbitrary pairs of locations (Krause et al., 2011b). Another application is in planning informative paths for mobile sensors (Singh et al., 2007). In these applications, greedy algorithms can be shown to perform arbitrarily poorly.

Calinescu and Zelikovsky (2005) develop a polynomial-time algorithm, which, given an integral-valued monotone submodular function f and a quota $0 \leq q \leq f(V)$, and any $\varepsilon > 0$, produce a tree of cost at most $O\Big(\frac{1}{\varepsilon}\frac{1}{\ln\ln n}(\ln n)^{2+\varepsilon}\log q\Big)$ times the optimal cost. For the related problem of path constraints, Chekuri and Pal (2005) develop an algorithm that, given a budget $B > 0$ and nodes $s, t \in V$ produces an s–t path of length at most B (if such exists) of submodular value $\Omega(\frac{OPT}{\log OPT})$. However, the running time of the algorithm is $(n \log B)^{O(\log n)}$, which is only quasi-polynomial in n. Nevertheless, Singh et al. (2007) show how this algorithm can be scaled to fairly large problems, and present results on planning informative paths for robotic sensors. For submodular functions that satisfy an additional *locality* property, which arises naturally in spatial monitoring problems, improved algorithms can be obtained, both for tree (Krause et al., 2006) and path (Singh et al., 2009) constraints.

3.4 Robust submodular optimization

In our example of placing sensors in a drinking water distribution network, we may wish to protect against malicious contaminations (Krause et al., 2008b). In this case, there may be a collection of m possible intrusion scenarios (e.g., locations whether contaminants could be introduced), and for each of the scenarios, we use a separate monotone submodular function $f_i(S)$ that quantifies the benefit (e.g., chance of detection) of having sensors at locations S, in scenario i. The problem of optimally placing sensors to protect against an adversary who wants to maximize their chances to go undetected therefore requires to solve

$$S^* = \arg\max_{|S|\leq k} \min_i f_i(S), \tag{14}$$

where $f_1, \ldots, f_m$ are monotone submodular functions.

Unfortunately, the function $f_{\min}(S) = \min_i f_i(S)$ is not generally submodular. Moreover, the greedy algorithm applied to $f_{\min}$ can perform arbitrarily poorly. In fact, Krause et al. (2008c) prove that Problem (14) is extremely inapproximable: Unless P=NP, *no* efficient algorithm can provide a solution S such that $f_{\min}(S) \geq \alpha(n)OPT_k$, where $OPT_k = \max_{|S'|\leq k} f_{\min}(S')$, for any function α that may even depend on the problem size n (for example, it is not efficiently possible to even recoup an exponentially small fraction of the optimal value). Perhaps surprisingly, given the hardness of the problem, it is possible to provide a different kind of approximation guarantee. Krause et al. (2008c) develop SATURATE, an algorithm that is guaranteed to efficiently obtain a set S such that $f_{\min}(S) \geq OPT_k$, and $|S| \leq \big(1 + \max_v \ln \sum_i f_i(\{v\})\big)k$. Thus, it is possible to obtain a set S that provides as much value as the best set of size k, at a cost that is logarithmically larger than k. This logarithmic approximation is optimal under reasonable complexity-theoretic assumptions. Problem 14 is much more general. For example, Schulman et al. (2011) have recently applied SATURATE in personal robotics in order to plan where to grasp objects.

Instead of committing to a fixed set $S \subseteq V$, in some applications it may be possible to select a probability distribution over sets. For example, when controlling a sensor network in a building to protect against intrusions, we may wish to obtain a randomized sensing strategy that performs as well as possible against an intruder who attempts to evade detection,

knowing the randomized strategy. This problem is formalized as

$$p^* = \operatorname*{arg\,max}_{p:p(S)>0\Rightarrow|S|\leq k} U(p) \text{ where } U(p) = \min_i \mathbb{E}_{S\sim p}[f_i(S)]\,.$$

Thus, we wish to obtain a distribution p^* over feasible sets S, such that our value is maximized in expectation, even under an adversarially chosen objective. Solving this problem optimally is a formidable task, as even representing the optimal distribution may require exponential space. However, Krause et al. (2011a) show how it is possible, for any $\varepsilon > 0$, to efficiently obtain a distribution $\hat{p}$ over $O(\ln m/\varepsilon^2)$ sets such that $U(\hat{p}) \geq (1-1/e)U(p^*) - \varepsilon$.

3.5 Nonmonotone submodular functions

While most work has focused on maximizing monotone submodular functions, several applications require maximizing *nonmonotone* submodular functions. For example, suppose we have a monotone submodular function f, and a modular cost function c, assigning each element v a cost $c(v)$. In this setting, we may wish to solve the unconstrained problem

$$\max_S f(S) - c(S).$$

Here, the function $g(S) = f(S) - c(S)$ is submodular, but nonmonotone. Another example is maximizing the symmetric mutual information $f(S) = I(\mathbf{X}_S; \mathbf{X}_{V\setminus S})$ (*c.f.*, §1.1). For arbitrary submodular functions f, even verifying whether there exists a set S such that $f(S) > 0$ is NP-hard (Feige et al., 2007), thus no approximation is possible. However, there have been recent breakthroughs on maximizing arbitrary *nonnegative* submodular functions (i.e., $f(S) \geq 0$ for all sets S).

Feige et al. (2007) prove that a local-search algorithm which iteratively adds or removes elements, ensuring that each addition or removal increases the function value by at least a multiplicative factor of $(1+\frac{\varepsilon}{n^2})$ terminates with a set S such that either S or $V\setminus S$ provides a $(\frac{1}{3} - \frac{\varepsilon}{n})$ approximation to the optimal unconstrained solution. They also prove that a more expensive, randomized local search procedure produces a solution with is a $(\frac{2}{5} - o(1))$ approximation to the optimal value. This is contrasted by their hardness result, proving that no approximation better than $\frac{1}{2}$ is achievable in the value oracle model. Krause and Horvitz (2008) present an application of unconstrained submodular maximization to the problem of trading off utility and privacy in online services.

Further improvements have been obtained utilizing the multilinear extension as discussed above. In particular, Gharan and Vondrák (2011) show that a simulated annealing algorithm is guaranteed to obtain a 0.41 approximation for unconstrained maximization, and 0.325 approximation for maximization subject to a matroid constraint. Most recently, Chekuri et al. (2011) show how the factor 0.325 can also be obtained for a constant number of knapsack constraints, and how a $0.19/k$ approximation is achievable for maximizing any nonnegative submodular function subject to k matroid and a constant number of knapsack constraints.

4 Online maximization of submodular functions

In the previous sections, we have assumed that we are given an objective function f that we wish to maximize. In some applications, the objective may not be known in advance. However, if we perform the same task repeatedly while facing objectives $f_1, \ldots, f_T$ drawn from some distribution, we might hope to learn to perform well on average over time. This is the premise behind *no–regret* algorithms, which are widely used in machine learning.

In many interesting applications the (unknown) objective functions f_t are monotone submodular. One example that we discuss below is learning to hybridize different algorithms for a computationally hard problem to generate a meta-algorithm which may outperform all of its constituent algorithms. Other examples include online sensor selection, news recommendation systems, online advertising and others. As we will see below, there are analogues of Theorem 1.5 in this no–regret setting, i.e., it is possible to learn to optimize submodular functions in an online manner.

4.1 The no–regret setting

Suppose we face the problem of repeatedly, over T rounds, choosing an action from a set V. In each round t, after selecting an action $v \in V$, you observe either the reward of every action (in the so-called *full information* feedback model) in that round, or merely the reward of the action you selected (in the so–called *bandit* feedback model). Let $r_t(v)$ denote the reward of action v in round t. Orthogonal to the feedback model, the rewards may be *stochastic*, in which case the reward functions $\{r_t : t = 1, 2, \ldots, T\}$ are drawn from some fixed (but unknown) distribution, or *non-stochastic*, in which case they may be arbitrary (even possibly chosen by an adversary). At first glance it seems impossible to give any interesting performance guarantees for the non-stochastic bandit setting. However, there are many beautiful results in this area based on the notion of minimizing the *regret* against the best action in hindsight (*c.f.*, Cesa-Bianchi and Lugosi 2006).

Definition 1.7 (Regret) The *regret* of action sequence $v_1, \ldots, v_T$ is

$$R_T = \max_{v^*} \sum_{t=1}^{T} r_t(v^*) - \sum_{t=1}^{T} r_t(v_t)$$

and the *average regret* is R_T/T.

There is a large literature on no–regret algorithms, to which we cannot do justice here. However, a key point is that if the reward functions are bounded, e.g., if $r_t(v) \in [0, 1]$ for all v and t, then in many settings, such as the case where V is finite, there are randomized algorithms whose expected regret grow as $o(T)$, so that the average regret R_T/T converges to zero as $T \to \infty$. There are algorithms which achieve $O(|V| \log T)$ expected regret for stochastic rewards with bandit feedback (Auer et al., 2002), $O(\sqrt{T \log |V|})$ for the non-stochastic rewards with full information feedback (Freund and Schapire, 1999), and $O\left(\sqrt{T|V| \log |V|}\right)$ for the non-stochastic rewards with bandit feedback (Auer et al., 2003). This means that in all settings we can converge to the performance of the best action in hindsight.

Now, suppose that instead of choosing an element $v \in V$ in each round you had to choose a set $S \subseteq V$ of bounded cardinality, say, $|S| \leq k$? Naïvely using the algorithms for the single action case ($k = 1$) by treating each feasible set as a distinct action has two major disadvantages. First, the algorithms require time at least linear in the number of feasible sets, namely $\binom{|V|}{k}$. Second, the average regret will shrink very slowly in the bandit feedback model, so that at least $\binom{|V|}{k}$ rounds are required before the regret bounds are meaningful. Unfortunately, for arbitrary reward functions $r_t(S)$, there is not much one can do about this situation. However, if the reward functions $r_t(\cdot)$ are monotone submodular, this structure can be exploited to get regret bounds roughly k times that of the original problem, with the caveat that the regret is against a $(1 - 1/e)$ approximation to the best feasible set in hindsight. We call such a regret measure the $(1 - 1/e)$-regret. Formally, for any $\alpha \geq 0$ the α-regret is defined as follows.

Definition 1.8 (α-Regret) The *α-regret* of a sequence of sets $S^{(1)}, \ldots, S^{(T)}$ is

$$R_\alpha = \alpha \cdot \max_{S^* \text{ feasible}} \sum_{t=1}^{T} r_t(S^*) - \sum_{t=1}^{T} r_t\Big(S^{(t)}\Big)$$

and the *average α-regret* is R_α/T.

4.2 Submodular maximization in the no–regret setting

Suppose $r_t : 2^V \to [0, 1]$ are monotone submodular, with bounded range. (We rescale so that the range is $[0, 1]$.) For clarity of exposition, we will focus here on a natural *partially transparent feedback model* defined as follows, but results in other models are available. For an ordered set S, let S_i be the first i elements of S. In the partially transparent feedback model, after selecting ordered set $S^{(t)}$, the marginal benefit of each action is revealed, assuming they were added in order. Formally, we observe $r_t\Big(S_i^{(t)}\Big) - r_t\Big(S_{i-1}^{(t)}\Big)$ for each $i = 1, 2, \ldots, k$. In this model, it is possible to obtain results like the following.

Theorem 1.9 (Streeter and Golovin 2007) *There is an efficient algorithm which incurs expected $(1 - 1/e)$-regret at most $O\left(k\sqrt{T|V|\log|V|}\right)$, in the non–stochastic setting with partially transparent feedback.*

Hence it is possible to ensure the expected average $(1 - 1/e)$-regret converges to zero at a rate proportional to $1/\sqrt{T}$, so that asymptotically the algorithm achieves at least $(1 - 1/e)$ of the optimal reward obtained by any fixed set of size k.

One algorithm obtaining the claimed regret bound, called the *online greedy* algorithm, combines the greedy algorithm with no-regret algorithms for the bandit feedback setting in a simple manner: There are k instantiations of such no-regret algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_k$, each with a set of actions V to choose among. In each round t, each $\mathcal{A}_i$ selects an action v_i^t and the set $S^{(t)} = \{v_1^t, \ldots, v_k^t\}$ is selected. For its trouble, $\mathcal{A}_i$ receives reward $r_t\Big(S_i^{(t)}\Big) - r_t\Big(S_{i-1}^{(t)}\Big)$, where $S_j^{(t)} = \{v_i^t : i \leq j\}$.

At a very high level, the key reason that the *online greedy* algorithm performs well is that the greedy algorithm is *noise-resistant* in the following sense. Suppose that instead

of selecting $S_i = S_{i-1} \cup \{\arg\max_v \Delta(v \mid S_{i-1})\}$ a "noisy" version of the greedy algorithm selects $S_i = S_{i-1} \cup \{v_i\}$ such that $\Delta(v_i \mid S_{i-1}) = \max_v \Delta(v \mid S_{i-1}) - \epsilon_i$ for some $\epsilon_i \geq 0$. Then a relatively straightforward modification of the proof by Nemhauser et al. (1978), which may be found in (Streeter and Golovin, 2008), shows that

$$f(S_k) \geq (1 - 1/e) \max_{S:|S|\leq k} f(S) - \sum_{i=1}^{k} \epsilon_i. \tag{15}$$

It turns out that the online greedy algorithm can be interpreted as a noisy version of the (standard) greedy algorithm running on a larger instance (which encodes all T instances the online greedy algorithm encounters), where the noise ϵ_i is exactly the regret of $\mathcal{A}_i$. Using known regret bounds for the case of selecting a single action then yields Theorem 1.9. Streeter and Golovin (2007) also provide results for other feedback models showing it is possible to obtain expected average $(1 - 1/e)$-regret which converges to zero as $T \to \infty$, up to and including the bandit feedback model in which only the reward of the set selected, namely $r_t\left(S_{i-1}^{(t)}\right)$, is observed at the end of round t (though in this very harsh feedback model the convergence rate on $(1 - 1/e)$-regret is much slower).

4.3 Applications of online maximization of submodular functions

There are several applications that can benefit from the results mentioned in §4.2. These include online sensor selection, database query optimization, news recommendation systems, online ad selection, and combining multiple heuristics for computationally hard problems (Streeter and Golovin, 2008; Golovin et al., 2010b; Munagala et al., 2005; Babu et al., 2004; Streeter et al., 2009; Radlinski et al., 2008; Streeter et al., 2007a). Here we discuss the problem of combining multiple heuristics in detail.

Combining multiple heuristics online Certain computationally hard problems such as integer programming and boolean satisfiability are ubiquitous in industrial and scientific applications, and a great deal of effort has gone into developing heuristics which solve them as quickly as possible in practice. Typically, there is no single heuristic which outperforms all others on every instance. Rather, heuristics often complement each other, so that there are opportunities for combining them into a meta-heuristic that outperforms its constituent heuristics in practice — in some cases by an order of magnitude or more. Before giving a concrete example where this effect occurs, we must clarify how heuristics are "combined." For this purpose, we use a *task switching schedule*:

Definition 1.10 (Run and Task Switching Schedule) For a heuristic h and $\tau \in \mathbb{R}_+$, a *run* is a pair (h, τ) representing the execution of h until either the problem is solved, or until τ time units have expired, after which execution is terminated. A *task switching schedule* is a sequence of runs, $\{(h_i, \tau_i)\}_{i\geq 0}$, and it's *length* is $\sum_{i\geq 0} \tau_i$.

A task switching schedule σ, such as the one illustrated in Figure 2, represents a meta-heuristic which performs its constituent runs in order and terminates immediately upon finding a solution. For example, the schedule $(h_1, 1), (h_2, 1)$ will first run h_1 until either the

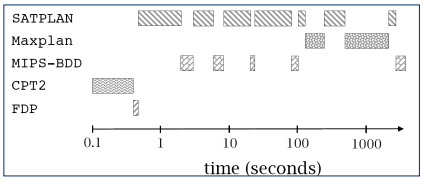


Figure 2 Illustration of a task switching schedule over five heuristics output by the online greedy algorithm.

solution is found or one time unit elapses, whichever comes first. If the first run terminates without finding the solution, the schedule next runs h_2 until either the solution is found or one time unit elapses, whichever comes first, and then terminates.

How is it that the best task switching schedule can outperform the best heuristic in it? A simple concrete example of this is for boolean satisfiability (SAT) where there are many randomized heuristics. Any fixed randomized heuristic h defines a family of deterministic heuristics consisting of the h run with random seed r, denoted by h_r, for each possible random seed. Empirically, the running time distribution of h with different seeds on an given SAT instance Φ is commonly heavy tailed. Hence it makes sense to periodically terminate h and restart it with a fresh random seed. In an example given by Streeter et al. (2007b), a particular heuristic solver had about a 20% chance of solving an instance after running for 2 seconds, but also a 20% chance that a run will not terminate after having run for 1000 seconds. Hence restarting the solver every 2 seconds rather than simply running it once until a solution is found reduces the mean time to solution by over an order of magnitude. When multiple solvers with complementary strengths are available, the potential for speedups increases. In experiments, when given access to state–of–the-art solvers entered into various academic competitions, the algorithm we are about to describe generated a meta-solver, which significantly outperformed all of the constituent solvers [5].

Two natural performance measures are *total execution time* to solve a set of instances, and *percent of instances solved* within a fixed time limit. The optimal task switching schedule will often outperform the best heuristic on both measures simultaneously. While there are strong empirical and theoretical results for both performance measures given by Streeter and Golovin (2008), for clarity of exposition here we focus on the latter.

Fix an arbitrary computational problem for which we have a finite set of (possibly randomized) heuristics H, and fix a time bound $B \in \mathbb{Z}_+$. We formulate the problem of combining heuristics as a problem of online maximization of submodular functions as follows. Construct a ground set $V = H \times \left\{\tilde{1}, \tilde{2}, \ldots, \tilde{B}\right\}$ of *potential-runs* in which $(h, \tilde{\tau}) \in V$ represents a random run which is the actual run (h, τ) with probability $1/\tau$ and is $(h, 0)$ (i.e., it does nothing) otherwise. A sequence of instances of our problem arrives online, one per round. In each round t the instance is represented by a monotone submodular function r_t which takes in a set of potential runs as input and gives the probability that at least one of them solves the instance. Formally,

$$r_t(S) \;:=\; 1 \;-\; \prod_{(h,\tilde{\tau})\in S} \left(1 - \frac{1}{\tau}\,\mathbb{P}\left[\text{run } (h,\tau) \text{ solves instance } t\right]\right). \tag{16}$$

Given a set S of potential runs of size B, we sample a task switching schedule by sampling their actual runs independently for each potential runs. The result is a set of runs R whose total length is B in expectation, and whose probability of solving the t^{th} instance is precisely $r_t(S)$ in expectation. Hence in each round t we can use the online greedy algorithm to select an ordered set of potential runs $S^{(t)}$, from which we can sample a task switching schedule $R^{(t)}$. Executing $R^{(t)}$ allows us to feedback unbiased estimates of the marginal increase in r_t from all of the potential runs, so that we are in the partially transparent feedback model. Hence we can generate a sequence $\left\{S^{(t)}\right\}_{t=1}^{T}$ that has vanishing average $(1-1/e)$-regret against any fixed set of potential runs. With slightly more work, one can show that the resulting sampled sequence of task switching schedules $\left\{R^{(t)}\right\}_{t=1}^{T}$ has vanishing average $(1-1/e)$-regret against any fixed task switching schedule of length B.

4.4 Online maximization with irrevocable choices: the submodular secretaries problem

In the *no–regret* setting of Section 4.1, the choices in any round are not constrained by what one did in previous rounds, and (typically) the goal is to perform well on average. By contrast, in the *competitive* online setting, one is faced with a sequence of irrevocable decisions among options that are revealed over time, and the goal is to do well in hindsight against any set of choices which were feasible given the revealed options. For example, in the *secretary problem*, you must hire exactly one applicant from a pool of applicants (of known size). Each applicant has a score, which is revealed during their interview. The applicants are interviewed in random order, and at the end of each interview you must irrevocably hire or reject the applicant being interviewed. The goal is to find a strategy maximizing the expected score of the hired applicant.

In *submodular secretary* problems, the applicants (denoted by the set V) are also interviewed in random order, and at the end of each interview you must irrevocably hire or reject the applicant being interviewed. However, now you are allowed to hire any subset of applicants in a feasible class $\mathcal{F} \subseteq 2^V$ (e.g., the independent sets of a matroid), and upon hiring a set S of applicants your reward is $f(S)$ for some nonnegative (possibly nonmonotone) submodular function f. Here, the function f is revealed to you during the interview process; after interviewing a set A of applicants, you are granted access to an oracle computing $f(S)$ for any $S \subseteq A$. The goal is to find a strategy maximizing the expected reward of the hired secretaries, measured with respect to f.

Submodular secretary problems generalize several important problems in practice, such as online bipartite matching (for e.g., matching display ads with search engine users) and certain caching problems. Gupta et al. (2010) and Bateni et al. (2010) provide constant competitive[6] algorithms for the uniform matroid (where $\mathcal{F} = \{S \subseteq V : |S| \leq k\}$ for some k). Gupta et al. also give a $O(\log r)$-competitive algorithm when $\mathcal{F}$ consists of the independent sets of a matroid of rank r, and Bateni et al. give an $O(\ell \log^2 r)$-competitive algorithm when $\mathcal{F}$ is the intersection of the independent sets of ℓ matroids of rank at most r.

5 Adaptive submodularity

In some applications we may wish to *adaptively* select a set, observing and taking into account feedback after selecting any particular element. For example, we may wish to sequentially activate sensors, adaptively taking into account measurements provided by the sensors selected so far when selecting the next sensor[7]. In such adaptive optimization problems, we must optimize over *policies*, i.e., functions from the information we have obtained to the next action. There are many variants of the problem, depending on which modeling assumptions (e.g., about the planning horizon, prior knowledge of the environment, and how the environment is affected by our actions) and goals (e.g., worst-case vs. average case reward) are suitable. Many such problems are notoriously intractable. In this section, we will review the notion of *adaptive submodularity* (Golovin and Krause, 2011b), a recent generalization of submodularity to adaptive optimization, that allows to develop efficient, provably near-optimal policies to an interesting class of adaptive optimization problems.

Example applications that exhibit adaptive submodular structure include problems in *active learning*, where we must adaptively select data points to label to maximize the performance of a classifier trained on the selected data points, *machine diagnosis*, where we must adaptively select tests to run on a patient or system to determine the best treatment plan, and certain *adaptive resource deployment* problems, where we irrevocably commit resources over time and may observe the benefits of our previous commitments before making additional commitments.

5.1 The adaptive submodularity framework

In order to formalize adaptive optimization, we need to describe the process of how we gather information. We model the state of the world abstractly as a random variable Φ, using ϕ to refer to a concrete value that Φ can take. In our sensing application, Φ may refer to the water quality at all nodes in the network. We presume a Bayesian model, so that we have a prior probability $\mathbb{P}[\phi]$ distribution over Φ. [8] We suppose there is a set of *actions* V we can perform and a set of *outcomes* O we might observe. We interpret the world state ϕ as a function from actions to outcomes of those actions, i.e., $\phi : V \to O$ and $\phi(v)$ is the outcome of performing action v. In our sensing application, $\phi(v)$ may refer to the particular measurement we obtain if we have a sensor at location v, and the world is in state ϕ. We represent the actions we have performed, as well as the outcomes we have observed as a partial function ψ from actions to outcomes. Hereby, the domain of ψ, denoted $\mathrm{dom}(\psi)$, is the set of actions performed up until that point. We call ϕ a *realization* (of the world-state) and ψ a *partial realization*. In our example, ψ may encode water quality measurements obtained at a subset $\mathrm{dom}(\psi)$ of nodes in the network. We assume there is an objective function $f : 2^V \times O^V \to \mathbb{R}_+$ indicating the reward $f(A, \phi)$ obtained from actions A under realization of the world state ϕ. A *policy* π can then be represented as a function from partial realizations ψ to the actions, so that $\pi(\psi)$ is the action taken by π upon observing ψ. See Figure 3 for an illustration. If ψ is not in the domain of π, then π terminates upon observing ψ. Finally, define $V(\pi, \phi)$ to be the set of actions played by π under realization ϕ. Informally, two natural optimization problems that arise are to get the most value out

of a fixed number of actions, and to get a certain amount of value with as few actions as possible. We formalize these as follows.

$\pi(\emptyset) = a$
$\pi(\{(a,0)\}) = b$
$\pi(\{(a,1)\}) = c$
$\pi(\{(a,0),(b,0)\}) = d$
$\pi(\{(a,0),(b,1)\}) = e$
$\pi(\{(a,1),(c,1)\}) = d$

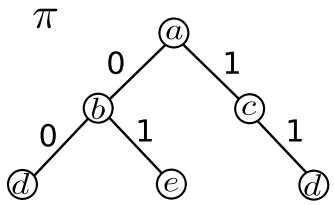


Figure 3 A policy and its representation as a decision tree.

Adaptive stochastic maximization. Based on the notation above, the expected reward of a policy π is

$$f_{\text{avg}}(\pi) := \mathbb{E}\left[f(V(\pi,\Phi),\Phi)\right] = \sum_{\phi} \mathbb{P}\left[\phi\right] f(V(\pi,\phi),\phi).$$

The goal of the *Adaptive Stochastic Maximization* problem is to find a policy π^* such that

$$\pi^* \in \arg\max_{\pi} f_{\text{avg}}(\pi) \text{ subject to } |V(\pi,\phi)| \leq k \text{ for all } \phi, \tag{17}$$

where k is a budget on how many actions can be played (e.g., we would like to adaptively choose k sensor locations such that the selected sensors provide as much information as possible in expectation).

Adaptive stochastic minimum cost cover. Alternatively, we can specify a quota q of reward that we would like to obtain, and try to find the cheapest policy achieving that quota (e.g., we would like to achieve a certain amount of information, as cheaply as possible in expectation). Formally, we define the average cost $c_{\text{avg}}(\pi)$ of a policy as the expected number of actions it plays, so that $c_{\text{avg}}(\pi) := \mathbb{E}\left[|V(\pi,\Phi)|\right]$. Our goal of this *Adaptive Stochastic Minimum Cost Cover* problem is then to find

$$\pi^* \in \arg\min_{\pi} c_{\text{avg}}(\pi) \text{ such that } f(V(\pi,\phi),\phi) \geq q \text{ for all } \phi, \tag{18}$$

i.e., the policy π^* that minimizes the expected number of items picked such that under all possible realizations, at least reward q is achieved.

Problems (17) and (18) are intractable in general, even to approximate to a factor of $O(|V|^{1-\epsilon})$, under reasonable complexity-theoretic assumptions. However, if f is satisfies certain conditions, which generalize monotonicity and submodularity, then the classic results bounding the performance of the greedy algorithm generalize. The following definitions are from (Golovin and Krause, 2011b), and presume that in order to observe ψ, all the actions in $\text{dom}(\psi)$ must have already been performed. They rely on a generalization of the discrete derivative $\Delta(v \mid S)$ to the adaptive setting.

Definition 1.11 (Conditional Expected Marginal Benefit) Given a partial realization ψ and an item v, the *conditional expected marginal benefit* of v conditioned on having observed ψ, denoted $\Delta(v \,|\, \psi)$, is

$$\Delta(v \,|\, \psi) := \mathbb{E}\left[f(\mathrm{dom}(\psi) \cup \{v\}, \Phi) - f(\mathrm{dom}(\psi), \Phi) \mid \psi\right] \tag{19}$$

where the expectation is taken with respect to $\mathbb{P}[\phi \mid \psi]$.

Definition 1.12 (Adaptive Monotonicity) A function $f : 2^V \times O^V \to \mathbb{R}_+$ is *adaptive monotone* with respect to distribution $\mathbb{P}[\phi]$ if the conditional expected marginal benefit of any item is nonnegative, i.e., for all ψ with $\mathbb{P}[\psi] > 0$ and all $v \in V$ we have

$$\Delta(v \,|\, \psi) \geq 0. \tag{20}$$

Definition 1.13 (Adaptive Submodularity) A function $f : 2^V \times O^V \to \mathbb{R}_+$ is *adaptive submodular* with respect to distribution $\mathbb{P}[\phi]$ if the conditional expected marginal benefit of any fixed item does not increase as more items are selected and their states are observed. Formally, f is adaptive submodular w.r.t. $\mathbb{P}[\phi]$ if for all ψ and ψ' such that ψ is a subrealization of ψ' (i.e., $\psi \subseteq \psi'$), and for all $v \in V \setminus \mathrm{dom}(\psi')$, we have

$$\Delta(v \,|\, \psi) \geq \Delta(v \,|\, \psi'). \tag{21}$$

Adaptive submodularity generalizes the classical notion of submodularity, in the sense that it reduces to submodularity in the case when the world state realization Φ is deterministic. The same is true for adaptive monotonicity. Not surprisingly, there is a natural generalization of the greedy algorithm as well, called the *adaptive greedy algorithm*, which iteratively selects the action maximizing the conditional expected marginal benefit, conditioned on the outcomes of all of its previous actions:

```
While not done
    Select v* ∈ arg max_v Δ(v | ψ);
    Observe φ(v*);
    Set ψ ← ψ ∪ {(v*, φ(v*))};
```
(22)

For adaptive stochastic maximization, the algorithm terminates after selecting k actions. For adaptive stochastic minimum cost cover, it stops when it has achieved the quota q of value, i.e., when it has observed ψ such that $f(\mathrm{dom}(\psi), \phi) \geq q$ for all $\psi \subseteq \phi$ (treating ϕ and ψ as relations, i.e., sets of input–output pairs).

Remarkably, it turns out that the adaptive greedy algorithm has performance guarantees that generalize various classic results for the greedy algorithm, for example, Theorem 1.5.

Theorem 1.14 (Golovin and Krause 2011b) *Let π_ℓ^{greedy} be the greedy policy implicitly represented by the pseudocode in (22), run for ℓ iterations (so that it selects ℓ actions), and let π_k^* be any policy selecting at most k actions for any realization ϕ. Then*

$$f_{avg}(\pi_\ell^{greedy}) \geq \left(1 - e^{-\ell/k}\right) f_{avg}(\pi_k^*) \tag{23}$$

where recall $f_{avg}(\pi) := \mathbb{E}[f(V(\pi, \Phi), \Phi)]$ is the expected reward of π.

Asadpour et al. (2008) prove Theorem 1.14 for a special case of stochastic submodular maximization. Golovin and Krause (2011b) also provide results for the adaptive stochastic min-cost cover problem (18) that generalize Theorem 1.6. Furthermore, Golovin and Krause (2011a) prove generalizations of results for maximizing monotone submodular functions under matroid constraints. Similarly, lazy evaluations (as discussed in §2) can still be applied to accelerate the adaptive greedy algorithm.

5.2 Example Applications

As mentioned in the beginning of this section, the adaptive submodularity framework has many applications. In some cases, greedily maximizing an adaptive submodular objective is already the algorithm of choice in practice. The framework then immediately provides theoretical justification in the form of approximation guarantees, and allows us to speed up existing algorithms. One example is *active learning* in the noiseless case (*c.f.*, Kosaraju et al. 1999; Dasgupta 2004; Golovin and Krause 2011b), in which we must adaptively select data points to be labelled for us (at some cost) until we can infer the labeling of all data points, while attempting to minimize the cost. In some other cases it is possible to frame the problem in the form of optimizing a carefully designed adaptive submodular objective. Given such an objective, the adaptive greedy algorithm may be used with this new objective to obtain a new approximation algorithm for the problem. Recent work on *active learning with noise* (Golovin et al., 2010a; Bellala and Scott, 2010), where the labels we receive may sometimes be incorrect, falls under this category. Active learning with noise can also be used to tackle *sequential experimental design* problems, in which an algorithm adaptively selects experiments to perform in order to distinguish scientific theories.

Another class of problems where adaptive submodularity is useful is for adaptively committing resources to achieve some goal. For example, in *adaptive sensor placement* one can deploy a set of sensors one by one, and decide where to place the next sensor based on the data obtained from previous sensors. In *adaptive viral marketing*, one must adaptively select people to target with a viral ad campaign – for example, to receive a free subscription to some service – on the premise that some of those targeted will enjoy the service, convince their friends to join, who will in turn convince their friends, and so on. For more information on these applications, see (Golovin and Krause, 2011b). Golovin et al. (2011) consider a *dynamic resource allocation* problem for conservation planning, in which a conservation agency with a fixed annual budget selects additional land parcels to buy each year while attempting to maximize the probability that certain (rare or endangered) species persist in the wild. Liu et al. (2008) consider a joint query optimization problem for streaming database systems; their problem is a special case of *stochastic set cover*, as discussed by Golovin and Krause (2011b).

5.3 Worst–Case Adaptive Optimization

Guillory and Bilmes (2010, 2011) provide a different recent approach to adaptive optimization based on submodularity, tailored to the worst–case scenario in which the outcome of any action is chosen adversarially. In their model there is a set of hypotheses H, actions V

with costs $c : V \to \mathbb{R}_+$, and outcomes O, as well as a set of monotone submodular functions $\{f_h : h \in H\}$ of type $2^{V \times O} \to \mathbb{R}_+$, and a threshold $\alpha \in \mathbb{R}_+$. Upon performing action v, an adversary selects outcome $o \in v(h^*)$, where h^* is an adversarially chosen hypothesis. The goal is to adaptively select, as cheaply as possible, actions until the resulting set of action–outcome pairs achieves α reward measured with respect to f_{h^*}. Guillory and Bilmes provide elegant reductions of this problem to a standard min-cost submodular cover problem, including one which, surprisingly, works even in the presence of certain types of adversarially selected noise. This allows them to obtain logarithmic approximations for these problems. They empirically tested their algorithms on a movie recommendation task in which users are asked a sequence of questions in an attempt to recommend movies for them to watch immediately.

6 Conclusions

We have reviewed the concept of submodular functions, a natural discrete analogue of convex functions. Focusing on the problem of maximizing submodular functions, we reviewed guarantees about efficient greedy methods, as well as more complex algorithms that can handle complex combinatorial constraints. We discussed extensions of submodular optimization to the online (no-regret and secretary) settings, as well as recent generalizations of submodularity to adaptive (interactive) optimization problems such as active learning.

While much progress was recently made, there are many interesting open problems, such as developing approximation algorithms for handling yet more general classes of constraints. In particular, the online- and adaptive extensions are still rather little explored. Lastly, while we focused on submodular maximization, there is a large literature on submodular minimization, with many open problems, such as the development of efficient methods for large-scale submodular minimization, and approximation algorithms for constrained minimization.

Submodularity is a broadly useful powerful concept, and we believe there are many interesting applications of submodular optimization to machine learning, AI and computer science in general yet to be discovered.

Notes

1 The study of submodular functions goes back at least to lattice theory (Bergmann, 1929). Edmonds (1970) first studied submodular functions in context of discrete optimization. See (Fujishige, 2005) and (Schrijver, 2003) for an in-depth discussion of submodular functions and their properties. This chapter focuses on modern results on submodular maximization.

2 There has been extensive research into algorithms for minimizing submodular functions (*c.f.*, Fujishige 2005; Schrijver 2003). Interestingly, unconstrained submodular minimization can be done efficiently, even if f can only be evaluated via a membership oracle (i.e., a black-box subroutine), whereas unconstrained maximization is NP-hard for general (non-monotone) submodular functions, since it includes the maximum cut problem as a special case (via the cut capacity example in Section 1.1). There are also efficient online algorithms for submodular minimization in the no-regret framework (Hazan and Kale, 2009; Jegelka and Bilmes, 2011b), which complement the results in Section 4.

3 The SFO toolbox for submodular optimization is available for download under `http://mloss.org/software/view/201/`

4 In some applications (*c.f.*, Kempe et al. 2003), calculating $f(S)$ may itself be difficult. In those cases, we may only be able to approximately evaluate $f(S)$ up to some multiplicative relative error α. Fortunately, most results about maximizing submodular functions are robust against such error (*c.f.*, Goundan and Schulz 2007; Calinescu et al. 2011; Streeter and Golovin 2008; Golovin and Krause 2011b).

5 Empirical evaluations of the online submodular maximization approach to combining heuristics appear in (Streeter et al., 2008) and (Streeter and Golovin, 2008). Additionally, a solver named MetaProver 1.0 developed based on these ideas competed in the SAT and FNT divisions of the 4th International Joint Conference on Automated Reasoning CADE ATP System Competition (`http://www.cs.miami.edu/~tptp/CASC/J4/`), where it won both divisions.

6 In the context of the submodular secretary problem, an algorithm is α-competitive if, in expectation over the random ordering of the interviews, it obtains at least α times the optimal value.

7 Note that in contrast with the secretary problems of Section 4.4, in the adaptive optimization problems of Section 5 the information we acquire depends on what action we select.

8 In general adaptive optimization problems, actions can alter the state of the world, however the framework described in Section 5 considers cases where the world state is a fixed sample from the distribution, and does not change in response to our actions.

References

Ageev, A. A., and Sviridenko, M. I. 2004. Pipage Rounding: a New Method of Constructing Algorithms with Proven Performance Guarantee. *Journal of Combinatorial Optimization*, **8**.

Asadpour, Arash, Nazerzadeh, Hamid, and Saberi, Amin. 2008. Stochastic Submodular Maximization. Pages 477–489 of: *WINE '08: Proc. of the 4th International Workshop on Internet and Network Economics*. Berlin, Heidelberg: Springer-Verlag.

Auer, Peter, Cesa-Bianchi, Nicolò, and Fischer, Paul. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, **47**(2), 235–256.

Auer, Peter, Bianchi, Nicolò, Freund, Yoav, and Schapire, Robert. 2003. The Nonstochastic Multiarmed Bandit Problem. *SIAM J. Comput.*, **32**(1), 48–77.

Babu, Shivnath, Motwani, Rajeev, Munagala, Kamesh, Nishizawa, Itaru, and Widom, Jennifer. 2004. Adaptive ordering of pipelined stream filters. Pages 407–418 of: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*.

Bateni, MohammadHossein, Hajiaghayi, MohammadTaghi, and Zadimoghaddam, Morteza. 2010. Submodular secretary problem and extensions. Pages 39–52 of: *Proceedings of the 13th international conference on Approximation, and 14 the International conference on Randomization,*

and combinatorial optimization: algorithms and techniques. APPROX/RANDOM'10. Berlin, Heidelberg: Springer-Verlag.

Bellala, G., and Scott, C. 2010. *Modified Group Generalized Binary Search with Near-Optimal Performance Guarantees*. Tech. rept. University of Michigan.

Bergmann, G. 1929. Zur Axiomatik der Elementargeometrie. *Monatshefte für Mathematik und Physik*, **36**, 269–284.

Birkhoff, G. 1933. On the Combination of Subalgebras. *Cambridge Philosophical Society*, **29**, 441–464.

Boykov, Yuri, and Jolly, Marie-Pierre. 2001. Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images. In: *International Conference on Computer Vision (ICCV)*.

Calinescu, G., and Zelikovsky, A. 2005. The Polymatroid Steiner Tree Problems. *Journal of Combinatorial Optimization*, **3**, 281–294.

Calinescu, Gruia, Chekuri, Chandra, Pal, Martin, and Vondrak, Jan. 2011. Maximizing a submodular set function subject to a matroid constraint. *to appear in SIAM Journal on Computing*.

Cesa-Bianchi, Nicolò, and Lugosi, Gábor. 2006. *Prediction, Learning, and Games*. Cambridge University Press.

Chekuri, Chandra, and Pal, Martin. 2005. A Recursive Greedy Algorithm for Walks in Directed Graphs. Pages 245–253 of: *FOCS*.

Chekuri, Chandra, Vondrák, Jan, and Zenklusen, Rico. 2011. Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes. In: *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*.

Dasgupta, Sanjoy. 2004. Analysis of a greedy active learning strategy. Pages 337–344 of: *NIPS: Advances in Neural Information Processing Systems*. MIT Press.

Edmonds, Jack. 1970. Submodular Functions, Matroids and Certain Polyhedra. In: *Combinatorial Structures and their Applications*. Gordon and Breach, New York.

Feige, U., Mirrokni, V., and Vondrak, J. 2007. Maximizing Non-monotone Submodular Functions. In: *FOCS*.

Feige, Uriel. 1998. A threshold of ln n for approximating set cover. *Journal of the ACM*, **45**(4), 634 – 652.

Freund, Y, and Schapire, RE. 1999. Adaptive Game Playing Using Multiplicative Weights. *Games and Economic Behavior*, **29**(1-2), 79–103.

Frieze, A. M. 1974. A cost function property for plant location problems. *Mathematical Programming*, **7**, 245–248. 10.1007/BF01585521.

Fujishige, S. 1978. Polymatroidal Dependence Structure of a Set of Random Variables. *Inform. Contr.*, **39**, 55–72.

Fujishige, Satoru. 2005. *Submodular functions and optimization*. 2nd edn. Vol. 58. North Holland, Amsterdam: Annals of Discrete Mathematics.

Gharan, Shayan Oveis, and Vondrák, Jan. 2011. Submodular Maximization by Simulated Annealing. In: *SODA*.

Goldengorin, Boris, Sierksma, Gerard, Tijssen, Gert A., and Tso, Michael. 1999. The Data-Correcting Algorithm for the Minimization of Supermodular Functions. *Management Science*, **45**(11), 1539–1551.

Golovin, Daniel, and Krause, Andreas. 2011a. Adaptive Submodular Optimization under Matroid Constraints. *CoRR*, **abs/1101.4450**.

Golovin, Daniel, and Krause, Andreas. 2011b. Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization. *Journal of Artificial Intelligence Research (JAIR)*. To appear.

Golovin, Daniel, Krause, Andreas, and Ray, Debajyoti. 2010a. Near-Optimal Bayesian Active

Learning with Noisy Observations. Pages 766–774 of: Lafferty, J., Williams, C. K. I., Shawe-Taylor, J., Zemel, R.S., and Culotta, A. (eds), *NIPS '10: Advances in Neural Information Processing Systems 23*.

Golovin, Daniel, Faulkner, Matthew, and Krause, Andreas. 2010b. Online Distributed Sensor Selection. In: *Proc. ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.

Golovin, Daniel, Krause, Andreas, Gardner, Beth, Converse, Sarah J., and Morey, Steve. 2011. Dynamic Resource Allocation in Conservation Planning. In: *AAAI '11: Proceedings of the Twenty–Fifth AAAI Conference on Artificial Intelligence*.

Goundan, Pranava R., and Schulz, Andreas S. 2007. *Revisiting the Greedy Approach to Submodular Set Function Maximization*. Working Paper. Massachusetts Institute of Technology.

Guillory, Andrew, and Bilmes, Jeff. 2010. Interactive Submodular Set Cover. In: *Proc. International Conference on Machine Learning (ICML)*.

Guillory, Andrew, and Bilmes, Jeff A. 2011. Simultaneous Learning and Covering with Adversarial Noise. In: *International Conference on Machine Learning (ICML)*.

Gupta, Anupam, Roth, Aaron, Schoenebeck, Grant, and Talwar, Kunal. 2010. Constrained Non-monotone Submodular Maximization: Offline and Secretary Algorithms. Pages 246–257 of: Saberi, Amin (ed), *WINE*. Lecture Notes in Computer Science, vol. 6484. Springer.

Hazan, Elad, and Kale, Satyen. 2009. Online Submodular Minimization. Pages 700–708 of: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A. (eds), *Advances in Neural Information Processing Systems 22*.

Jegelka, Stefanie, and Bilmes, Jeff. 2011a. Submodularity Beyond Submodular Energies: Coupling Edges in Graph Cuts. In: *CVPR*.

Jegelka, Stefanie, and Bilmes, Jeff A. 2011b. Online Submodular Minimization for Combinatorial Structures. In: *International Conference on Machine Learning (ICML)*.

Kawahara, Yoshinobu, Nagano, Kiyohito, Tsuda, Koji, and Bilmes, Jeff. 2009. Submodularity Cuts and Applications. In: *NIPS*.

Kempe, David, Kleinberg, Jon, and Tardos, Éva. 2003. Maximizing the spread of influence through a social network. Pages 137–146 of: *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM.

Kohli, Pushmeet, Kumar, Pawan, and Torr, Philip. 2009. $\mathcal{P}^3$ & Beyond: Move Making Algorithms for Solving Higher Order Functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **31**(9), 1645 – 1656.

Kosaraju, S. Rao, Przytycka, Teresa M., and Borgstrom, Ryan S. 1999. On an Optimal Split Tree Problem. Pages 157–168 of: *Proc. of the 6th Intl. Workshop on Algorithms and Data Structures*. London, UK: Springer-Verlag.

Krause, A., and Guestrin, C. 2005. Near-optimal Nonmyopic Value of Information in Graphical Models. In: *Proc. of Uncertainty in Artificial Intelligence (UAI)*.

Krause, A., and Horvitz, E. 2008. A Utility-Theoretic Approach to Privacy and Personalization. In: *Proc. 23rd Conference on Artificial Intelligence (AAAI), Special Track on AI & the Web*.

Krause, A., Guestrin, C., Gupta, A., and Kleinberg, J. 2006. Near-optimal Sensor Placements: Maximizing Information while Minimizing Communication Cost. In: *Proceedings of the Fifth International Symposium on Information Processing in Sensor Networks (IPSN)*.

Krause, A., Singh, A., and Guestrin, C. 2008a. Near-optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. In: *Journal of Machine Learning Research*, vol. 9.

Krause, Andreas. 2010. SFO: A Toolbox for Submodular Function Optimization. *Journal of Machine Learning Research (JMLR)*, **11**, 1141–1144.

Krause, Andreas, and Guestrin, Carlos. 2007. Near-optimal observation selection using submodular functions. Pages 1650–1654 of: *AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence*. AAAI Press.

Krause, Andreas, Leskovec, Jure, Guestrin, Carlos, VanBriesen, Jeanne, and Faloutsos, Christos. 2008b. Efficient Sensor Placement Optimization for Securing Large Water Distribution Networks. *Journal of Water Resources Planning and Management*, **134**(6), 516–526.

Krause, Andreas, McMahan, Brendan, Guestrin, Carlos, and Gupta, Anupam. 2008c. Robust Submodular Observation Selection. *Journal of Machine Learning Research (JMLR)*, **9**(December), 2761–2801.

Krause, Andreas, Rajagopal, Ram, Gupta, Anupam, and Guestrin, Carlos. 2009. Simultaneous Placement and Scheduling of Sensors. In: *In Proc. ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.

Krause, Andreas, Roper, Alex, and Golovin, Daniel. 2011a. Randomized Sensing in Adversarial Environments. In: *IJCAI '11: Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. To appear.

Krause, Andreas, Guestrin, Carlos, Gupta, Anupam, and Kleinberg, Jon. 2011b. Robust Sensor Placements at Informative and Cost-Effective Locations. *ACM Transactions on Sensor Networks*, **7**(4).

Kulik, A., Shachnai, H., and Tamir, T. 2009. Maximizing submodular functions subject to multiple linear constraints. In: *Proc. of ACM-SIAM SODA*.

Leskovec, Jure, Krause, Andreas, Guestrin, Carlos, Faloutsos, Christos, VanBriesen, Jeanne, and Glance, Natalie. 2007. Cost-effective outbreak detection in networks. Pages 420–429 of: *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM.

Lin, Hui, and Bilmes, Jeff. 2011. A Class of Submodular Functions for Document Summarization. In: *In North American chapter of the Association for Computational Linguistics/Human Language Technology Conference (NAACL/HLT-2011)*.

Liu, Zhen, Parthasarathy, Srinivasan, Ranganathan, Anand, and Yang, Hao. 2008. Near-optimal algorithms for shared filter evaluation in data stream systems. Pages 133–146 of: *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM.

Lovasz, L. 1983. Submodular functions and convexity. *Mathematical Programming - State of the Art*, 235–257.

Minoux, M. 1978. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques, LNCS*, 234–243.

Munagala, Kamesh, Babu, Shivnath, Motwani, Rajeev, Widom, Jennifer, and Thomas, Eiter. 2005. The pipelined set cover problem. Pages 83–98 of: *Proceedings of the International Conference on Database Theory*.

Narasimhan, Mukund, and Bilmes, Jeff. 2004. PAC-learning bounded tree-width Graphical Models. In: *Uncertainty in Artificial Intelligence*.

Narasimhan, Mukund, Jojic, Nebojsa, and Bilmes, Jeff. 2005. Q-clustering. In: *NIPS*.

Nemhauser, G. L., and Wolsey, L. A. 1978. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Research*, **3**(3), 177–188.

Nemhauser, G. L., and Wolsey, L. A. 1981. Maximizing submodular set functions: formulations and analysis of algorithms. *Studies on Graphs and Discrete Programming, volume 11 of Annals of Discrete Mathematics*.

Nemhauser, George L., Wolsey, Laurence A., and Fisher, Marshall L. 1978. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, **14**(1), 265–294.

Radlinski, Filip, Kleinberg, Robert, and Joachims, Thorsten. 2008. Learning Diverse Rankings with Multi-Armed Bandits. Pages 784–791 of: *ICML*.

Schrijver, Alexander. 2003. *Combinatorial optimization : polyhedra and efficiency*. Volume B, Part IV, Chapters 39-49. Springer.

Schulman, John D., Goldberg, Ken, and Abbeel, Pieter. 2011. Grasping and Fixturing as Submodular Coverage Problems. In: *ISRR*.

Singh, Amarjeet, Krause, Andreas, Guestrin, Carlos, Kaiser, William J., and Batalin, Maxim A. 2007 (January). Efficient Planning of Informative Paths for Multiple Robots. Pages 2204–2211 of: *International Joint Conference on Artificial Intelligence (IJCAI)*.

Singh, Amarjeet, Krause, Andreas, and Kaiser, William. 2009. Nonmyopic Adaptive Informative Path Planning for Multiple Robots. In: *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*.

Stobbe, Peter, and Krause, Andreas. 2010. Efficient Minimization of Decomposable Submodular Functions. In: *Proc. Neural Information Processing Systems (NIPS)*.

Streeter, Matthew, and Golovin, Daniel. 2007. *An Online Algorithm for Maximizing Submodular Functions*. Tech. rept. CMU-CS-07-171. Carnegie Mellon University.

Streeter, Matthew, and Golovin, Daniel. 2008. An Online Algorithm for Maximizing Submodular Functions. Pages 1577–1584 of: *NIPS*.

Streeter, Matthew, Golovin, Daniel, and Smith, Stephen F. 2007a. Combining Multiple Heuristics Online. Pages 1197–1203 of: *AAAI '07: Proceedings of the Twenty–Second AAAI Conference on Artificial Intelligence*. Menlo Park, California: AAAI Press.

Streeter, Matthew, Golovin, Daniel, and Smith, Stephen F. 2007b. Restart Schedules for Ensembles of Problem Instances. Pages 1204–1210 of: *AAAI '07: Proceedings of the Twenty–Second AAAI Conference on Artificial Intelligence*. Menlo Park, California: AAAI Press.

Streeter, Matthew, Golovin, Daniel, and Smith, Stephen F. 2008. Combining Multiple Constraint Solvers: Results on the CPAI'06 Competition Data. Pages 11–18 of: *Proceedings of the Second International CSP Solver Competition*.

Streeter, Matthew, Golovin, Daniel, and Krause, Andreas. 2009. Online Learning of Assignments. Pages 1794–1802 of: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A. (eds), *NIPS '09: Advances in Neural Information Processing Systems 22*.

Sviridenko, M. 2004. A Note on Maximizing a Submodular Set Function Subject to Knapsack Constraint. *Operations Research Letters*, **32**, 41–43.

Vondrák, Jan. 2008. Optimal approximation for the submodular welfare problem in the value oracle model. Pages 67–74 of: *STOC*.

Vondrák, Jan. 2010. Submodularity and curvature: the optimal algorithm. *RIMS Kokyuroku Bessatsu*, **B23**, 253–266.

Wolsey, Laurence A. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, **2**(4), 385–393.