

Chapter 2

Centralized Submodular Optimization

This chapter gives an overview of algorithms for solving problems of the form

$$\begin{array}{ll} \text{maximize} & f(S) \\ & S \subseteq V \\ \text{s.t.} & S \in \mathcal{C} \end{array} \quad (2.1)$$

where $f : 2^V \rightarrow \mathbb{R}$ is a submodular function and $\mathcal{C}$ is a collection of feasible sets, by a centralized entity that can compute $f(S)$ for any set $S \subseteq V$. The advantages of submodularity for problems of the form (2.1) are twofold. First, submodularity is inherent in a variety of application domains. Second, submodular structures lead to polynomial time algorithms with provable optimality bounds for solving a variety of discrete optimization problems.

A set of motivating applications that exhibit the structure (2.1) is presented first. Cardinality-constrained submodular maximization is considered, including classical greedy algorithms, optimality bounds, and worst-case complexity results. A worst-case version of cardinality-constrained submodular maximization is considered next. The techniques in the cardinality-constrained case are generalized to matroid constraints, as well as online submodular maximization.

2.1 Applications of Submodular Maximization

This section discusses motivating applications of submodular maximization, including sensor placement, maximizing influence in social networks, and document summarization.

2.1.1 Sensor Placement

Low-cost wireless sensors are widely deployed for uses including structural health monitoring, water quality inspection, analyzing road traffic patterns, and wildlife monitoring. The sensor placement problem consists of selecting a set of sensor locations in order to maximize the information gathered by the sensors. The submodular structure of this problem was first identified in [17].

Let V denote the set of possible sensor locations, and define $A \subseteq V$ to be the set of locations that are occupied by sensors. Each location $v \in V$ is assumed to be associated with a random variable X_v . Define $X_A = \{X_v : v \in A\}$. One metric for evaluating the quality of sensor placement is the *mutual information* between the known variables X_A and the unknown variables $X_{V \setminus A}$ [17]:

$$F(A) = I(X_A; X_{V \setminus A}). \quad (2.2)$$

Intuitively, Eq. (2.2) represents the amount of information from the unknown variables $X_{V \setminus A}$ that can be inferred from the known variables X_A .

Lemma 2.1 *The function $F(A)$ is submodular as a function of A .*

Proof The first step is to show that the function $g(A) = H(X_A)$, where $H(X_A)$ denotes the joint entropy of the random variables $\{X_v : v \in A\}$, is submodular as a function of A . Using the identity $H(X, Y) = H(X) + H(Y|X)$, for any $A \subseteq B$ and $v \notin B$, the increments are given by

$$\begin{aligned} g(A \cup \{v\}) - g(A) &= H(X_{A \cup \{v\}}) - H(X_A) = H(X_v | X_A) \\ &\geq H(X_v | X_B) = H(X_{B \cup \{v\}}) - H(X_B) = g(B \cup \{v\}) - g(B), \end{aligned}$$

where the inequality follows from the fact that conditioning reduces entropy [6].

Now, $F(A)$ can be written as

$$F(A) = I(X_A; X_{V \setminus A}) = H(X_A) + H(X_{V \setminus A}) - H(X_V) = g(A) + g(V \setminus A) - g(V),$$

which is a sum of submodular functions and hence is submodular.

2.1.2 Influence in Social Networks

Diffusion of information in social networks plays an important role in scientific innovation and viral marketing. In order to maximize the spread or influence of a particular idea, one approach is to implant the idea into a subset of seed users, who then influence their neighbors through word of mouth [7]. For a set of seed users A , define the function

$$F(A) = \text{expected number of users adopting the idea from seed users } A.$$

The structure of $F(A)$ depends on the information diffusion model that is used. In [14], it was proved that, for the *threshold model* (in which a user adopts an idea if a threshold number of neighbors adopts the idea) and the *cascade model* (in which each user succeeds in influencing each neighbor with a certain probability), the function $F(A)$ is submodular as a function of A . The problem of selecting a set of up to k seed users to maximize adoption of an idea, denoted $\max \{F(A) : |A| \leq k\}$, is therefore a submodular maximization problem.

2.1.3 Document Summarization

Document summarization can be viewed as an optimization problem by denoting the set of sentences in the documents as $V = \{s_1, \dots, s_n\}$, and attempting to find a subset of sentences S that maximizes the information content. Many of the utility functions used in document summarization have been shown to exhibit submodularity [20, 21]. For example, for the *maximum marginal relevance* (MMR) metric [4], the gain from adding an element s_k to a set S is equal to

$$\lambda \text{Sim}_1(s_k, q) - (1 - \lambda) \max_{i \in S} \text{Sim}_2(s_i, s_k).$$

$\text{Sim}_1(s_k, q)$ represents the similarity between sentence s_k and query q , $\text{Sim}_2(s_i, s_k)$ denotes the similarity between sentences s_i and s_k , and $\lambda \in [0, 1]$.

An additional feature of the document summarization problem is that, since different sentences will have different length, some sentences may be more costly than others. Letting c_i denote the cost (length) of each sentence and b denote the total budget for the summary, the document summarization problem can be formulated as

$$\begin{aligned} & \text{maximize } MMR(S) \\ & S \subseteq V \\ & \text{s.t.} \quad \sum_{i \in S} c_i \leq b \end{aligned}$$

This is a submodular maximization problem with a *knapsack constraint*.

2.2 Cardinality-Constrained Submodular Maximization

The cardinality-constrained submodular maximization problem is formulated as

$$\begin{aligned} & \text{maximize } f(S) \\ & \text{s.t.} \quad |S| \leq k \end{aligned} \tag{2.3}$$

where $f(S)$ is a submodular function. The following greedy algorithm is known to provide provable approximation guarantees for (2.3).

Algorithm 2.1 Algorithm for cardinality-constrained submodular maximization.

```

1: procedure GREEDY_SUBMODULAR( $f(S), k$ )
2:   Input: Submodular function  $f(S)$ 
3:   Cardinality constraint  $k$ 
4:   Output: Set  $S$ 
5:   Initialization:  $S \leftarrow \emptyset, i \leftarrow 0$ 
6:   while  $i \leq k$  do
7:      $s_i^* \leftarrow \arg \max_{i \in V \setminus S} \{f(S \cup \{s_i^*\}) - f(S)\}$ 
8:     if  $f(S) - f(S \cup \{s_i^*\}) \leq 0$  then
9:       return  $S$ 
10:    else
11:       $S \leftarrow S \cup \{s_i^*\}$ 
12:       $i \leftarrow i + 1$ 
13:    end if
14:  end while
15:  return  $S$ 
16: end procedure

```

At each iteration of the greedy algorithm, the element s_i^* that maximizes the increment $f(S \cup \{s_i^*\}) - f(S)$ is added to the set S , terminating when $|S| = k$. The following theorem, first proved in [23], characterizes the optimality bound of Algorithm 2.1.

Theorem 2.1 *Let S^* denote the optimal solution to (2.3). If the function $f(S)$ is monotone and satisfies $f(\emptyset) = 0$, then the set S returned by Algorithm 2.1 satisfies*

$$f(S) \geq (1 - 1/e) f(S^*).$$

Proof Let S_i denote the set selected by Algorithm 2.1 at iteration i . The first step is to show that, at each iteration, there exists a such that

$$f(S_i \cup \{a\}) - f(S_i) \geq \frac{1}{k}(f(S^*) - f(S_i)).$$

By monotonicity,

$$f(S^*) - f(S_i) \leq f(S^* \cup S_i) - f(S_i).$$

Writing $S^* \cup S_i = S_i \cup \{a_1, \dots, a_l\}$ yields the telescoping sum

$$f(S^* \cup S_i) - f(S_i) = \sum_{j=1}^l (f(S_i \cup \{a_1, \dots, a_j\}) - f(S_i \cup \{a_1, \dots, a_{j-1}\})).$$

Submodularity and the above two identities then implies

$$f(S^*) - f(S_i) \leq \sum_{j=1}^l (f(S_i \cup \{a_j\}) - f(S_i))$$

and there exists $a^* \in S^* \setminus S_i$ such that

$$f(S^*) - f(S_i) \leq |S^* \setminus S_i| (f(S_i \cup \{a^*\}) - f(S_i)) \leq k(f(S_i \cup \{a^*\}) - f(S_i)).$$

Since $f(S_{i+1}) - f(S_i) \geq f(S_i \cup \{a\}) - f(S_i)$ for any a , we have

$$f(S_{i+1}) - f(S_i) \geq \frac{1}{k}(f(S^*) - f(S_i)).$$

Rearranging terms and subtracting both sides from $f(S^*)$ yields

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

Applying this inequality k times implies that

$$f(S^*) - f(S) \leq \left(1 - \frac{1}{k}\right)^k (f(S^*) - f(\emptyset)) \leq \frac{1}{e} f(S^*),$$

and hence $f(S) \geq (1 - 1/e) f(S^*)$.

The greedy algorithm can also be shown to provide the best possible optimality guarantees in the worst case, unless $P = NP$.

Theorem 2.2 *If there exists a polynomial time algorithm that returns a set $\hat{S}$ satisfying $f(\hat{S}) \geq (1 - 1/e + \varepsilon) f(S^*)$ for any instance of (2.3) and $\varepsilon > 0$, then $P = NP$.*

Theorem 2.2 was first shown in [22] by constructing a family of submodular functions that serve as counterexamples. A reduction to the known NP-hard problem max-3SAT(5) was demonstrated in [8].

Example 2.1 The set cover problem is a special case of submodular maximization. Let $A = \{a_1, \dots, a_m\}$ be a finite set, and define a set of subsets $A_1, \dots, A_n \subseteq A$. Define $V = \{1, \dots, m\}$ and

$$f(S) = \left| \bigcup_{i \in S} A_i \right|.$$

As shown in Chap. 1 (Example 1.2), the function $f(S)$ is submodular as a function of S . Hence, the problem of selecting a collection of subsets $\{A_i : i \in S\}$ in order to maximize $f(S)$ can be formulated as a submodular maximization problem.

Consider the set $A = \{a_1, a_2, \dots, a_6\}$, and define $A_1 = \{a_1, a_2\}$, $A_2 = \{a_1, a_4\}$, $A_3 = \{a_2, a_3\}$, $A_4 = \{a_1, a_6\}$, and $A_5 = \{a_2, a_5\}$. Let $k = 2$. By following the greedy algorithm, we first set $S = \{1\}$, with $f(S) = 2$. All remaining elements will then satisfy $f(S \cup \{v\}) - f(S) = 1$, so let $S = \{1, 2\}$ with $f(S) = 3$. Note that S is not the optimal solution; we have $f(\{2, 3\}) = 4$, for example. However, we have

$$3 = f(S) \geq \left(1 - \frac{1}{e}\right) 4 = \left(1 - \frac{1}{e}\right) f(S^*),$$

verifying that the $(1 - 1/e)$ bound is satisfied.

A dual formulation for Eq. (2.3) is given by

$$\begin{aligned} & \text{minimize } |S| \\ & \text{s.t.} \quad f(S) \geq \alpha \end{aligned} \tag{2.4}$$

for a constant α . A greedy algorithm for Eq. (2.4) is shown as Algorithm 2.2.

Algorithm 2.2 Algorithm for selecting the minimum-size set to achieve a bound on a submodular function.

```

1: procedure GREEDY_SUBMODULAR_DUAL( $f(S)$ ,  $\alpha$ )
2:   Input: Submodular function  $f(S)$ 
3:   Desired upper bound  $\alpha$ 
4:   Output: Set  $S$ 
5:   Initialization:  $S \leftarrow \emptyset$ 
6:   while  $f(S) < \alpha$  do
7:      $s_i^* \leftarrow \arg \max_{i \in V \setminus S} \{f(S \cup \{s_i^*\}) - f(S)\}$ 
8:     if  $f(S) - f(S \cup \{s_i^*\}) \leq 0$  then
9:       return  $S$ 
10:    else
11:       $S \leftarrow S \cup \{s_i^*\}$ 
12:    end if
13:  end while
14:  return  $S$ 
15: end procedure

```

Algorithm 2.2 adds the element v to S that maximizes $f(S \cup \{v\}) - f(S)$ at each iteration until $f(S) \geq \alpha$. Before analyzing the optimality of Algorithm 2.2, we have the following preliminary result:

Lemma 2.2 ([28]) *Let $0 \leq u_1 \leq \dots \leq u_n$ and $x_1 \geq x_2 \geq \dots \geq x_n > 0$. If*

$$S = \sum_{i=1}^{n-1} (u_i(x_i - x_{i+1})) + u_n x_n = u_1 x_1 + \sum_{i=1}^{n-1} (u_{i+1} - u_i) x_{i+1},$$

then

$$S \leq \left(\max_i u_i x_i \right) \left[1 = \log \left\{ \min \left(\frac{x_1}{x_n}, \frac{u_n}{u_1} \right) \right\} \right].$$

The optimality guarantees of Algorithm 2.2 are described as follows:

Theorem 2.3 ([28]) *Let S^* denote the optimal solution to (2.4), and let S denote the set returned by Algorithm 2.2. Then*

$$\frac{|S|}{|S^*|} \leq 1 + \ln \left\{ \frac{f(V) - f(\emptyset)}{f(V) - f(\hat{S})} \right\},$$

where $\hat{S}$ is the value of the set S at the second-to-last iteration of Algorithm 2.2.

Proof The approach of the proof is to consider a linear programming relaxation of (2.4), given as

$$\begin{aligned} & \text{minimize} \quad \sum_j y_j \\ & y_1, \dots, y_n \\ & \text{s.t.} \quad \sum_{j=1}^n (f(S_i \cup \{j\}) - f(S_i)) y_j \geq f(V) - f(S_i), \quad i = 0, \dots, (n-1) \\ & \quad y_j \geq 0 \end{aligned} \tag{2.5}$$

Equation (2.5) provides a lower bound on (2.4), and hence any dual feasible solution to (2.5) is also a lower bound on Eq. (2.4). Define vector $\mathbf{u}^i$ by $u_j^i = \frac{1}{\theta^i}$ if $i = t$ and 0 otherwise, where $\theta_t = \min_{j \in V \setminus S_{t-1}} \left\{ \frac{1}{f(S_{t-1} \cup \{j\}) - f(S_{t-1})} \right\}$. Then $\mathbf{u}_i$ is a dual feasible solution of (2.5), and hence

$$\begin{aligned} & \max_{t=1, \dots, n} \{u_t(f(V) - f(\emptyset), \dots, f(V) - f(S_{t-1}))\} \\ & = \max_{t=1, \dots, T} \{\theta_t(f(V) - f(S_{t-1}))\} \leq |S^*|. \end{aligned} \tag{2.6}$$

Furthermore, by Lemma 2.2,

$$\begin{aligned} |S| &= \sum_{t=1}^{k-1} [\theta_t(f(S_t) - f(S_{t-1}))] + \theta_k(f(V) - f(S_{k-1})) \\ &\leq \max_t \{\theta_t(f(V) - f(S_{t-1}))\} \left[1 + \log \left(\frac{f(V) - f(\emptyset)}{f(V) - f(S_{T-1})} \right) \right]. \end{aligned} \tag{2.7}$$

Combining (2.6) and (2.7) yields

$$|S| \leq |S^*| \left[1 + \log \left(\frac{f(V) - f(\emptyset)}{f(V) - f(S_{n-1})} \right) \right],$$

as desired.

The results of Theorem 2.3 imply that efficient approximation algorithms can be developed for problems such as selecting a minimum-size set of sensors to provide a given level of information. The dual algorithm is also a basic building block of robust submodular maximization, described as follows.

2.3 Robust Submodular Maximization

The *robust submodular maximization* problem can be motivated as follows. Consider the sensor placement problem defined in Sect. 2.1.1. While the set of possible sensor locations V is known *a priori*, the probability distribution of the measured data $\{X_v : v \in V\}$ may be uncertain. For example, there may be a set of possible correlation matrices $\Sigma_1, \dots, \Sigma_M$, leading to a set of possible objective functions $f_1, \dots, f_M$. In order to maximize performance under these conditions, the goal is to maximize the *worst-case* objective function, leading to the formulation

$$\begin{aligned} & \text{maximize } \min_{i=1, \dots, M} f_i(S) \\ & \text{s.t. } |S| \leq k \end{aligned} \quad (2.8)$$

Equation (2.8) defines the robust submodular maximization problem. A key obstacle is the fact that the minimum of a collection of submodular functions is not necessarily submodular, as shown in the following example.

Example 2.2 Let $V = \{1, 2\}$ and define functions $f_1 : 2^V \rightarrow \mathbb{R}$ and $f_2 : 2^V \rightarrow \mathbb{R}$ as

$$\begin{aligned} f_1(\emptyset) &= 0 & f_2(\emptyset) &= 0 \\ f_1(\{1\}) &= 3 & f_2(\{1\}) &= 1 \\ f_1(\{2\}) &= 1 & f_2(\{2\}) &= 2 \\ f_1(\{1, 2\}) &= 3.5 & f_2(\{1, 2\}) &= 2.5 \end{aligned}$$

By inspection, f_1 and f_2 are monotone submodular functions. Define $F(S) = \min \{f_1(S), f_2(S)\}$. Then $F(\emptyset) = 0$, $F(\{1\}) = 1$, $F(\{2\}) = 1$, and $F(\{1, 2\}) = 2.5$. Thus $F(\{1, 2\}) - F(\{1\}) = 1.5$ and $F(\{2\}) - F(\emptyset) = 1$, implying that the definition of submodularity does not hold.

Unlike maximization of a submodular function, there is no polynomial time algorithm for robust submodular maximization with any provable guarantee, unless $P = NP$, as shown in the following proposition:

Proposition 2.1 ([18]) *Let $\gamma : \mathbb{N} \rightarrow \mathbb{R}_{>0}$. Define S^* to be the optimal solution to (2.8), and suppose that there exists an algorithm that returns a set S satisfying $F(S) > \gamma(n)F(S^*)$. Then $P = NP$.*

Proof Let $\{A_i : i = 1, \dots, M\}$ be a collection of subsets of V . The *hitting set problem* is to find the smallest subset S such that $S \cap A_i \neq \emptyset$, for all $i = 1, \dots, M$. The hitting set problem is known to be NP-hard. For each i , define function f_i as $f_i(S) = 1$ if $S \cap A_i \neq \emptyset$ and $f_i(S) = 0$ otherwise. The function f_i is submodular and monotone nondecreasing.

Now, let k denote the size of the minimum-size subset of S with $S \cap A_i \neq \emptyset$. Then the problem

$$\begin{aligned} & \text{maximize } \min_{i=1, \dots, M} f_i(S) \\ & \text{s.t. } |S| \leq k \end{aligned} \quad (2.9)$$

has optimum value 1, and the optimal solution is a hitting set. Suppose that there exists an algorithm satisfying the conditions of the proposition. Then the algorithm will return a set S satisfying $F(S) \geq \gamma(n) > 0$, implying that $F(S) = 1$ since $F(S) \in \{0, 1\}$. Thus there exists a polynomial time algorithm that finds a hitting set, establishing that $P = NP$.

In [18], an approximation algorithm was proposed that achieves a weaker notion of optimality. The algorithm relies on the fact that the dual optimization problem

$$\begin{aligned} & \text{minimize } |S| \\ & \text{s.t.} \quad \min_{i=1, \dots, n} f_i(S) \geq \alpha \end{aligned} \quad (2.10)$$

is equivalent to the submodular optimization problem

$$\begin{aligned} & \text{minimize } |S| \\ & \text{s.t.} \quad \frac{1}{M} \sum_{i=1}^M \min \{f_i(S), \alpha\} \geq \alpha \end{aligned} \quad (2.11)$$

The submodular structure holds because the function $g(S) = \min \{f(S), \alpha\}$ is submodular for any monotone submodular function $f(S)$ and constant α (Lemma 1.4). Hence the approximation algorithm selects a parameter α , and then approximately solves (2.11) to find a minimum-size set S satisfying $F(S) \geq \alpha$. If $|S| \leq \beta k$ for some parameter β , then the value of α is increased. Otherwise, α is decreased in order to find a set S satisfying the desired constraints. A detailed description is given as Algorithm 2.3.

The optimality guarantee of Algorithm 2.3 is as follows:

Algorithm 2.3 Robust submodular maximization algorithm.

```

1: procedure ROBUST_SUBMODULAR( $f_1(S), \dots, f_M(S), k, \delta$ )
2:   Input: Submodular functions  $f_1(S), \dots, f_M(S)$ 
3:   Cardinality constraint  $k$ , parameter  $\delta > 0$ 
4:   Output: Set  $S$ 
5:   Initialization:  $\alpha_{\min} \leftarrow 0, \alpha_{\max} \leftarrow F(V), S^* \leftarrow \emptyset$ 
6:   while  $(\alpha_{\max} - \alpha_{\min}) > \delta$  do
7:      $\alpha \leftarrow \frac{\alpha_{\max} + \alpha_{\min}}{2}, S \leftarrow \emptyset$ 
8:     while  $\hat{F}(S) \leq \alpha$  do
9:        $v^* \leftarrow \arg \max \{\hat{F}(S \cup \{v\}) - \hat{F}(S) : v \in V\}$ 
10:       $S \leftarrow S \cup \{v^*\}$ 
11:    end while
12:    if  $|S| \leq \beta k$  then
13:       $S^* \leftarrow S, \alpha_{\min} \leftarrow \alpha$ 
14:    else
15:       $\alpha_{\max} \leftarrow \alpha$ 
16:    end if
17:  end while
18:  return  $S^*$ 
19: end procedure

```

Lemma 2.3 *Algorithm 2.3 returns a set S satisfying $F(S) \geq F(S^*)$ and $|S| \leq \beta k$ when*

$$\beta = 1 + \log \left(\max_{s \in V} \sum_i f_i(s) \right)$$

and δ is sufficiently small.

Proof By Theorem 2.3, the inner while loop will result in a set S with

$$\frac{|S|}{|\hat{S}|} \leq 1 + \log \left(\max_{s \in V} \sum_{i=1}^M f_i(s) \right),$$

where

$$\hat{S} = \arg \min \{ |S| : \hat{F}(s) \geq \alpha \}.$$

For δ sufficiently small, α will be approximately equal to $F(S^*)$. Hence the set S returned by the algorithm will satisfy $F(S) \geq F(S^*)$ and $|S| \leq \beta |S^*| = \beta k$.

Example 2.3 An example of robust submodular maximization in the influence maximization application of Sect. 2.1.2 is as follows. For this application, define the objective function $f(S)$ to be equal to the expected number of nodes that are influenced by the seed nodes S under the *independent cascade model* [12] in a social network G . The objective function is equal to the number of nodes that are connected to at least one node in S in a randomly generated subgraph G' , with each link included in the subgraph with probability q . The graph topology of G is assumed to be an *Erdos–Renyi* graph, in which a link exists between two nodes i and j with probability p , independent of all other links. The parameter $p = 0.1$, $q = 0.8$, and the number of nodes is given by $N = 30$. The number of inputs varies from 1 to 8. For the robust maximization problem, the objective functions $f_1(S), f_2(S), \dots, f_M(S)$ correspond to independently generated realizations of G .

The degree of influence over the network is shown as Fig. 2.1. The robust submodular maximization approach is compared with selecting random seed nodes, and provides a higher level of influence over the network in the worst case. On average, roughly half of the network nodes are influenced by the robust submodular maximization approach with three seed nodes.

2.4 Submodular Maximization with Matroid Constraints

The problem of maximizing a submodular function subject to matroid constraints is formulated as

$$\begin{aligned} & \text{maximize } f(S) \\ & \text{s.t.} \quad S \in \mathcal{M} \end{aligned} \tag{2.12}$$

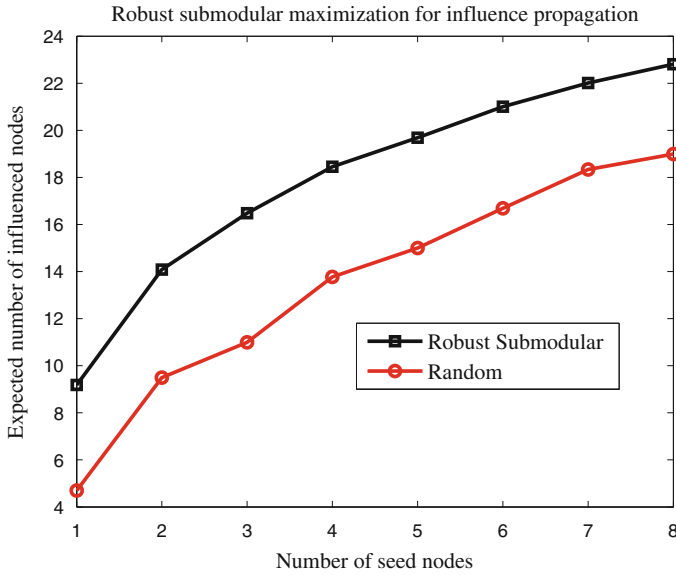


Fig. 2.1 Case study of robust influence maximization. A set of k seed nodes is selected in order to maximize the worst-case influence over the network, based on three possible topologies. The network topologies are chosen as i.i.d. Erdos-Renyi random graphs, with 30 nodes and a link probability of 0.1. Influence propagation is based on the independent cascade model [12]

where $\mathcal{M} = (V, \mathcal{I})$ is a matroid. The constraint $\{|S| \leq k\}$ is a special case of a matroid constraint when the matroid is uniform.

Equation (2.12) can also be solved via a greedy approach analogous to Algorithm 2.1. Under the modified algorithm, Step 7 is replaced with

$$s_i^* \leftarrow \arg \max \{f(S \cup \{s_i^*\}) - f(S) : (S \cup \{s_i^*\}) \in \mathcal{M}\}.$$

In the special case where the function f is modular, the greedy algorithm is optimal.

Theorem 2.4 ([24]) *If $f(S)$ is a modular function, i.e., $f(S) = \sum_{i \in S} c_i$ for some real numbers $c_1, \dots, c_n$, then the greedy algorithm returns the optimal solution to (2.12).*

Proof Assume that $c_i > 0$, for all $i \in S$; otherwise, the ground set of $\mathcal{M}$ can be restricted to $\{i : c_i > 0\}$. Define $S = \{s_1, \dots, s_k\}$ to be the set chosen by the greedy algorithm, with $c_{s_1} \geq c_{s_2} \geq \dots \geq c_{s_k}$. Let $B = \{b_1, \dots, b_k\}$ be another basis of $\mathcal{M}$ with $c_{b_1} \geq c_{b_2} \geq \dots \geq c_{b_k}$. The approach is to show that $c_{s_i} > c_{b_i}$, for all i .

Suppose not, and let i be the smallest integer with $c_{s_i} < c_{b_i}$. Define $S_i = \{s_1, \dots, s_{i-1}\}$ and $B_i = \{b_1, \dots, b_i\}$. Since $|S_i| < |B_i|$, there exists $r \in \{1, \dots, i\}$ such that $(S_i \cup \{b_r\}) \in \mathcal{M}$. By definition of the b_i 's, $c_{b_r} \geq c_{b_i} > c_{s_i}$. Hence the greedy algorithm should choose element b_r at the i th iteration, creating a contradiction. This

contradiction implies that $c_{s_i} \geq c_{b_i}$, for all i , and hence $f(S) \geq f(B)$, for all bases B of $\mathcal{M}$. Since the c_i 's are assumed to be positive, the maximizers of $f(S)$ must be bases, and hence $f(S) \geq f(A)$, for all $A \in \mathcal{M}$.

The greedy algorithm, however, does not achieve the same optimality guarantees as in the cardinality-constrained case for arbitrary submodular functions.

Proposition 2.2 ([11]) *Let S^* denote the optimal solution to (2.12). The greedy algorithm returns a set S satisfying $f(S) \geq \frac{1}{2}f(S^*)$.*

Proof Let $S^* \setminus S = \{a_1, \dots, a_r\}$. By monotonicity of $f(S)$,

$$f(S^*) \leq f(S \cup S^*) = f(S) + \sum_{j=1}^r (f(S \cup \{a_1, \dots, a_j\}) - f(S \cup \{a_1, \dots, a_{j-1}\})).$$

Submodularity implies that each term can be bounded above as

$$f(S^*) \leq f(S) + \sum_{j=1}^r (f(S \cup \{a_j\}) - f(S)) \leq f(S) + \sum_{j=1}^r (f(S_{j-1} \cup \{a_j\}) - f(S_{j-1})),$$

where S_{j-1} is the set S at the j th iteration of the greedy algorithm. Letting s_j^* denote the element selected by the greedy algorithm at step j , by definition we have

$$f(S_{j-1} \cup \{a_j\}) - f(S_{j-1}) \leq f(S_{j-1} \cup \{s_j^*\}) - f(S_{j-1}).$$

Substituting this inequality yields

$$\begin{aligned} f(S^*) &\leq f(S) + \sum_{j=1}^r (f(S_{j-1} \cup \{a_j\}) - f(S_{j-1})) \\ &\leq f(S) + \sum_{j=1}^k (f(S_{j-1} \cup \{a_j\}) - f(S_{j-1})) = 2f(S), \end{aligned}$$

completing the proof.

Proposition 2.2 implies that the greedy algorithm achieves a worst-case optimality gap of $1/2$, which is less than the cardinality-constrained bound of $(1 - 1/e)$. The optimality bound in the matroid-constrained case can be improved via the *continuous greedy algorithm*, described as follows:

The continuous greedy algorithm was first proposed in [3] and consists of two stages. In the first stage, a continuous optimization problem $\max \{F(y) : y \in \mathcal{P}(\mathcal{M})\}$ is solved, where $F(y)$ is a continuous extension of the submodular function $f(S)$ and $\mathcal{P}(\mathcal{M})$ is a relaxation of the matroid constraint $S \in \mathcal{M}$. In the second stage, the obtained continuous solution y is rounded to obtain a discrete set S .

The function $F : [0, 1]^n \rightarrow \mathbb{R}$ is the *multilinear relaxation* of $f(S)$, defined by

$$F(y) = \sum_{R \subseteq V} \left[f(R) \prod_{i \in R} y_i \prod_{i \notin R} (1 - y_i) \right]. \quad (2.13)$$

The multilinear extension can be interpreted as computing $\mathbf{E}(f(\mathbf{S}))$, where $\mathbf{S}$ is a random variable obtained by sampling each element $i \in S$ independently with probability y_i . As a special case, if y_i is a 0-1 vector with $S = \{i : y_i = 1\}$, then $F(y) = f(S)$. The following lemma gives basic properties of $F(y)$.

Lemma 2.4 *If $f(S)$ is monotone and submodular, then the function F satisfies the following properties:*

- F has second partial derivatives everywhere,
- For every i , $\frac{\partial F}{\partial y_j} \geq 0$ everywhere (monotonicity),
- For every i, j , $\frac{\partial^2 F}{\partial y_i \partial y_j} \leq 0$ everywhere (continuous submodularity).

Proof The existence of second partial derivatives is clear by inspection. Differentiating with respect to y_j yields

$$\frac{\partial F}{\partial y_j} = \sum_{R \subseteq V \setminus \{j\}} [(f(R \cup \{j\}) - f(R))] \geq 0,$$

implying monotonicity.

The second derivatives satisfy

$$\begin{aligned} & \frac{\partial^2 F}{\partial y_i \partial y_j} \\ &= \sum_{R \subseteq V \setminus \{i, j\}} \left[(f(R \cup \{i, j\}) - f(R \cup \{i\}) - f(R \cup \{j\}) + f(R)) \prod_{l \in R} y_l \prod_{l \notin R} (1 - y_l) \right]. \end{aligned}$$

Since

$$\prod_{l \in R} y_l \prod_{l \notin R} (1 - y_l) \geq 0$$

for $y \in [0, 1]^n$, and

$$f(R \cup \{i, j\}) - f(R \cup \{i\}) \leq f(R \cup \{j\}) - f(R)$$

by submodularity, $\frac{\partial^2 F}{\partial y_i \partial y_j} \leq 0$, for all $i \neq j$. When $i = j$, $\frac{\partial^2 F}{\partial y_i^2} = 0$.

Definition 2.1 Let $\mathcal{M} = (V, \mathcal{I})$ be a matroid. For any set $I \subseteq V$, define $\mathbf{1}_I$ to be a vector with $(\mathbf{1}_I)_i = 1$ if $i \in I$ and $(\mathbf{1}_I)_i = 0$ otherwise. The *matroid polytope* $\mathcal{P}(\mathcal{M})$ is defined by

$$\mathcal{P}(\mathcal{M}) \triangleq \text{conv}\{\mathbf{1}_I : I \in \mathcal{I}\}, \quad (2.14)$$

where conv denotes the convex hull of a set. The *base polytope* $\mathcal{B}(\mathcal{M})$ is defined by

$$\mathcal{B}(\mathcal{M}) \triangleq \text{conv}\{\mathbf{1}_I : I \text{ is a basis of } \mathcal{I}\}. \quad (2.15)$$

The continuous greedy algorithm is a continuous analog to Algorithm 2.1. The vector y is initialized to 0. At each iteration, a vector v satisfying

$$v = \arg \max \{(\nabla F(y))^T v : v \in \mathcal{P}(\mathcal{M})\}$$

is found, so that v is a steepest ascent direction for $F(y)$. The vector y is incremented to $y + \varepsilon v$ for some small $\varepsilon > 0$. The continuous stage of the algorithm terminates when y reaches the base polytope of $\mathcal{M}$. In the second stage, the continuous solution y is rounded to a discrete set S .

Algorithm 2.4 Algorithm for cardinality-constrained submodular maximization.

```

1: procedure CONTINUOUS_GREEDY_SUBMODULAR( $f(S), \mathcal{M}$ )
2:   Input: Submodular function  $f(S)$ 
3:   Matroid  $\mathcal{M} = (V, \mathcal{I})$ 
4:   Output: Set  $S$ 
5:   Initialization:  $k \leftarrow \text{rank of } \mathcal{M}, y(0) \leftarrow 0, t \leftarrow 0$ 
6:   while  $t < 1$  do
7:     Define  $R(t)$  as random variable taking each  $i \in V$  independently with probability  $y_i(t)$ 
8:     for  $i = 1, \dots, n$  do
9:        $\omega_i(t) \leftarrow \mathbf{E}(f(R(t) \cup \{j\}) - f(R(t)))$ 
10:    end for
11:     $I(t) \leftarrow \arg \max \{\sum_{i \in I} \omega_i(t) : I \in \mathcal{M}\}$ 
12:     $y(t + \delta) \leftarrow y(t) + \delta \mathbf{1}_{I(t)}$ 
13:     $t \leftarrow t + \delta$ 
14:  end while
15:   $S \leftarrow \text{PipageRound}(y, \mathcal{M})$ 
16:  return  $S$ 
17: end procedure

```

The pipage rounding routine called by Algorithm 2.4 is described in [3]. The optimality guarantees provided by the continuous phase of the algorithm are described as follows:

Lemma 2.5 *For any $y \in [0, 1]^n$, let R denote a random set where $i \in R$ with probability y_i . Let S^* denote the optimal solution to (2.12). Then*

$$f(S^*) \leq F(y) + \max_{I \in \mathcal{M}} \sum_{j \in I} \mathbf{E}(f(R \cup \{j\}) - f(R)) \quad (2.16)$$

Proof By submodularity,

$$f(S^*) \leq f(R) + \sum_{j \in S^*} (f(R \cup \{j\}) - f(R))$$

for any set R . By taking the expectation over R , we have

$$f(S^*) \leq F(y) + \sum_{j \in S^*} \mathbf{E}(f(R \cup \{j\}) - f(R)) \leq F(y) + \sum_{j \in I} \mathbf{E}(f(R \cup \{j\}) - f(R)).$$

Theorem 2.5 *For any $\varepsilon > 0$, δ can be chosen sufficiently small such that the fractional solution y obtained by Algorithm 2.4 satisfies*

$$F(y) \geq \left(1 - \frac{1}{e} - \frac{\varepsilon}{k}\right) f(S^*). \quad (2.17)$$

Proof Define $\Delta_j(t) = y_j(t + \delta) - y_j(t) = \delta \mathbf{1}_{I(t)}$, and let $D(t)$ be a random subset of $I(t)$ where each element j appears independently with probability $\Delta_j(t)$. We have that

$$F(y(t + \delta)) = \mathbf{E}[f(R(t + \delta))] \geq \mathbf{E}[f(R(t) \cup D(t))],$$

since the probability of each element being included in $R(t + \delta)$ is greater than the probability of each element being included in $R(t) \cup D(t)$. Hence

$$\begin{aligned} F(y(t + \delta)) - F(y(t)) &\geq \mathbf{E}[f(R(t) \cup D(t)) - f(R(t))] \\ &\geq \sum_j [\Pr(D(t) = \{j\}) \mathbf{E}(f(R(t) \cup \{j\}) - f(R(t)))] \\ &\geq \delta \sum_{j \in I(t)} \mathbf{E}(f(R(t) \cup \{j\}) - f(R(t))) \end{aligned}$$

By Lemma 2.5,

$$F(y(t + \delta)) - F(y(t)) \geq \delta(f(S^*) - F(y(t))),$$

implying that

$$f(S^*) - F(y(m\delta)) \leq (1 - \delta)^m f(S^*)$$

at time $t = m\delta$. When $t = 1$ and hence $m = 1/\delta$,

$$f(S^*) - F(y(1)) \leq (1 - \delta)^{1/\delta} f(S^*) \leq \frac{1}{e} f(S^*).$$

Rearranging terms yields (2.17).

The first stage of the continuous greedy algorithm results in a relaxed fractional solution y satisfying $F(y) \geq (1 - \frac{1}{e}) f(S^*)$. The second stage of the algorithm rounds the fractional solution to obtain a set S satisfying $S \in \mathcal{M}$ and $F(y) = f(S)$, thus preserving the optimality guarantees of the first (continuous) stage of the algorithm.

2.5 Online Submodular Maximization

In *online submodular maximization*, the goal is to maximize a time-varying set of submodular functions $\{f_t(S_t) : t = 1, 2, \dots, T\}$. At the beginning of time t , the objective function f_t is unknown, while the objective functions for previous time periods $f_1, \dots, f_{t-1}$ are known. The goal is to select the sets $S_1, \dots, S_T$ as solutions to

$$\begin{aligned} & \text{maximize } \frac{1}{T} \sum_{t=1}^T f_t(S_t) \\ & \text{s.t. } |S_t| \leq k \end{aligned} \quad (2.18)$$

The problem formulation (2.18) arises in settings including influence maximization in dynamic social networks, sensor network scheduling for random processes that change distribution over time, and scheduling computing tasks over a time horizon [25].

Online submodular maximization algorithms have been proposed that achieve analogous optimality guarantees to Theorem 2.1 [25], through online versions of Algorithm 2.1. Since the function $f_t(S_t)$ is unknown, the optimal node to select at each iteration of the greedy algorithm is estimated using *experts algorithms*. As a preliminary, background on experts algorithms is given as follows.

2.5.1 Background on Experts Algorithms

Experts algorithms are techniques for predicting and learning an optimal action over time [5]. In an experts algorithm, a user takes advice from a set of K experts $\{e_1, \dots, e_K\}$. Since the advice may be contradictory, the user selects a single expert at time step m and follows that advice. After making a choice $e[m]$ at step m , the user is given a payoff $\ell_m(e[m])$, as well as the payoffs $\ell_m(e_1), \dots, \ell_m(e_K)$ that would have been received from following each other expert. Based on this feedback, the user updates its probabilities of following the advice of each expert at the next time step. The goal of the user is to choose $e[1], \dots, e[M]$ to maximize $\sum_{m=1}^M \ell_m(e[m])$.

It is a known result that any algorithm for selecting $e[1], \dots, e[M]$ can provide arbitrarily poor performance compared to the optimal selection $e[1]^*, \dots, e[M]^*$. On the other hand, there are efficient algorithms that provide provable guarantees compared to the best *fixed* expert, i.e.,

$$\sum_{m=1}^M \ell_m(e[m]) \approx \max_{r=1, \dots, K} \frac{1}{M} \sum_{m=1}^M \ell_m(e_r).$$

One such algorithm is the *exponentially weighted average* (EWA) algorithm. Under this algorithm, the user maintains a weight w_r^m for each action $e_1, \dots, e_K$ at iteration m . The action $e[m]$ is chosen at time m from a probability distribution obtained by normalizing r , i.e.,

$$Pr(e[m] = e_r) = \frac{w_r^m}{\sum_{j=1}^K w_j^m}.$$

After receiving the payoffs $\ell_m(e_r)$, $r = 1, \dots, K$ at step m , the weights are updated according to an exponential rule, with the new weight w_r^{m+1} given by $w_r^{m+1} = \beta^{\ell_m(e_r)} w_r^m$ for some $\beta \in (0, 1]$.

Intuitively, this approach assigns a higher weight to actions that have received a larger payoff during previous time periods. The following theorem provides an optimality bound for EWA.

Theorem 2.6 *The payoff achieved by the actions $e[1], \dots, e[M]$ selected by EWA satisfies*

$$\sum_{m=1}^M \ell_m(e[m]) + o(M) \geq \max_{r=1, \dots, K} \sum_{m=1}^M \ell_m(e_r). \quad (2.19)$$

The property of (2.19) is referred to as *Hannan consistency*.

2.5.2 Online Submodular Maximization Algorithm

An algorithm for online submodular maximization is presented as Algorithm 2.5. The algorithm maintains a set of k instances of the experts algorithm EWA, with each instance corresponding to a single iteration of the greedy algorithm. Each of these k instances of the algorithm consists of n experts, with expert j corresponding to choosing element j . The weight w_{ij} corresponds to the benefit from choosing element j at the i th iteration. Hence, for each time t , the algorithm proceeds over k iterations. At the i th iteration, each weight w_{ij} is updated using the benefit $f(S_{t-1, i-1} \cup \{j\}) - f(S_{t-1, i-1})$. An element is then chosen from a probability distribution derived by normalizing the weights; hence the algorithm is performing a randomized version of greedy selection using the experts algorithm.

The following theorem describes the optimality gap of Algorithm 2.5.

Theorem 2.7 *The sets $S_1, \dots, S_T$ selected by Algorithm 2.5 satisfy*

$$\frac{1}{T} \sum_{t=1}^T f_t(S_t) + o(T) \geq (1 - 1/e) \frac{1}{T} \sum_{t=1}^T f_t(S^*) \quad (2.20)$$

where

$$S^* = \arg \max \left\{ \frac{1}{T} \sum_{t=1}^T f_t(S) : |S| \leq k \right\}.$$

Proof The proof follows the proof of Theorem 2.1. Let $S_{t,i}$ denote the set S_t at the i th iteration of the greedy algorithm, and let $S^* \setminus S_{t,i} = \{a_1^t, \dots, a_j^t\}$. Then

Algorithm 2.5 Algorithm for online submodular maximization.

```

1: procedure ONLINE_SUBMODULAR( $f_1, \dots, f_T, k$ )
2:   Input: Set of submodular functions  $f_1, \dots, f_T$  where  $f_{t-1}$  is revealed at time  $t$ , cardinality
      constraint  $k$ 
3:   Output: Set  $S_1, \dots, S_T$ 
4:   Initialization:  $w_{ij} \leftarrow 1, i = 1, \dots, k, j = 1, \dots, n$ 
5:   for  $t = 1, \dots, T$  do
6:     Function  $f_{t-1}$  revealed at time  $t$ 
7:     for  $i = 1, \dots, k$  do
8:       for  $j = 1, \dots, n$  do
9:          $w_{i,j} \leftarrow \beta^{f_t(S_{t-1,i-1} \cup \{j\}) - f_t(S_{t-1,i-1})} w_{i,j}$ 
10:      end for
11:       $\mathbf{p}_i \leftarrow \mathbf{w}_i / (\mathbf{1}^T \mathbf{w}_i)$ 
12:    end for
13:     $S_t \leftarrow \emptyset$ 
14:    for  $i = 1, \dots, k$  do
15:      Select an element  $s_i$  according to probability distribution  $\mathbf{p}_i$ 
16:       $S_t \leftarrow S_t \cup \{s_i\}$ 
17:    end for
18:  end for
19:  return  $S_1, \dots, S_T$ 
20: end procedure

```

$$\begin{aligned}
\frac{1}{T} \sum_{t=1}^T (f_t(S^*) - f_t(S_{t,i})) &\leq \frac{1}{T} \sum_{t=1}^T (f_t(S_{t,i} \cup S^*) - f_t(S_{t,i})) \\
&= \frac{1}{T} \sum_{t=1}^T \sum_j (f_t(S_{t,i} \cup \{a_1^t, \dots, a_j^t\}) - f_t(S_{t,i} \cup \{a_1^t, \dots, a_{j-1}^t\}))
\end{aligned}$$

holds by monotonicity. Submodularity implies that

$$\frac{1}{T} \sum_{t=1}^T (f_t(S^*) - f_t(S_{t,i})) \leq \frac{1}{T} \sum_{t=1}^T \sum_j (f_t(S_{t,i} \cup \{a_j^t\}) - f_t(S_{t,i})).$$

Hence, there exists an element a_i^* such that

$$\frac{1}{T} \sum_{t=1}^T (f_t(S^*) - f_t(S_{t,i})) \leq \frac{k}{T} \sum_{t=1}^T (f_t(S_{t,i} \cup \{a_i^*\}) - f_t(S_{t,i})).$$

Now, by the Hannan consistency property of the experts algorithm, the elements $\{s_{t,i} : t = 1, \dots, T\}$ satisfy

$$\frac{1}{T} \sum_{t=1}^T (f_t(S_{t,i-1} \cup \{s_{t,i}\}) - f_t(S_{t,i})) + o(T) \geq \frac{1}{T} \sum_{t=1}^T (f_t(S_{t,i-1} \cup \{a\}) - f_t(S_{t,i-1}))$$

for any fixed element a . Rearranging terms yields

$$\frac{1}{T} \sum_{t=1}^T (f_t(S^*) - f_t(S_{t,i+1})) - o(T) \leq \left(1 - \frac{1}{k}\right) \sum_{t=1}^T (f_t(S^*) - f_t(S_{t,i})).$$

Applying this bound at all of the k iterations yields (2.20).

Analogous to the experts algorithm of Sect. 2.5.1, the optimality guarantees provided by Algorithm 2.5 are in comparison with the best possible fixed set S . The proof of the optimality bound can be viewed as consisting of two components. The first component shows the local optimality of each iteration i , which approximates the guarantee $f(S_{i-1} \cup \{v_i^*\}) \geq f(S_{i-1} \cup \{v\})$, for all $v \in V$ that is inherent to the greedy algorithm. The second component exploits local optimality, submodularity, and monotonicity to prove a global optimality bound.

2.6 Summary and Notes

This section gave an overview of centralized submodular maximization techniques. After providing motivating applications of submodular maximization, cardinality-constrained submodular maximization algorithms were presented. It was proved that the greedy algorithm achieves a $(1 - 1/e)$ optimality gap, and that this is the best achievable gap unless $P = NP$. Robust submodular maximization was discussed, and it was shown that no algorithm achieves a provable optimality gap unless $P = NP$. A greedy heuristic was then presented and analyzed. The more general problem of submodular maximization with matroid constraints was treated, and it was shown that the greedy algorithm is optimal for modular functions and $\frac{1}{2}$ -optimal for monotone submodular functions. A continuous greedy algorithm that improves the optimality gap to $(1 - 1/e)$ was then presented. Finally, online submodular maximization, in which a set of elements must be chosen based on past observations of the objective function, was discussed.

The greedy algorithm for submodular maximization was first presented and analyzed in [23]. Recent breakthroughs have improved the optimality bounds for submodular maximization under constraints, including one or more matroids [3, 10, 19], knapsacks [27], and submodular knapsacks [13], as well as submodular maximization without constraints [2, 9]. The online submodular maximization problem has been studied in [1, 26]. For a survey on submodular maximization, see [16]. Other applications of submodular maximization include image segmentation [15] and data collection [29].

References

1. Bateni, M., Hajiaghayi, M., Zadimoghaddam, M.: Submodular secretary problem and extensions. *ACM Trans. Algorithms (TALG)* **9**(4), 32 (2013)
2. Buchbinder, N., Feldman, M., Naor, J.S., Schwartz, R.: A tight linear time $(1/2)$ -approximation for unconstrained submodular maximization. In: *IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 649–658 (2012)
3. Calinescu, G., Chekuri, C., Pal, M., Vondrak, J.: Maximizing a submodular set function subject to a matroid constraint. *SIAM J. Comput.* **40**(6), 1740–1766 (2011)
4. Carbonell, J., Goldstein, J.: The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: *Proceedings of the ACM SIGIR*, pp. 335–336 (1998)
5. Cesa-Bianchi, N., Lugosi, G.: *Prediction, Learning, and Games*. Cambridge University Press, Cambridge (2006)
6. Cover, T.M., Thomas, J.A.: *Elements of Information Theory*. Wiley, New York (2012)
7. Domingos, P., Richardson, M.: Mining the network value of customers. In: *Proceedings of the Seventh ACM SIGKDD International Conference On Knowledge Discovery and Data Mining*, pp. 57–66 (2001)
8. Feige, U.: A threshold of $\ln n$ for approximating set cover. *J. ACM (JACM)* **45**(4), 634–652 (1998)
9. Feige, U., Mirrokni, V.S., Vondrak, J.: Maximizing non-monotone submodular functions. *SIAM J. Comput.* **40**(4), 1133–1153 (2011)
10. Filmus, Y., Ward, J.: Monotone submodular maximization over a matroid via non-oblivious local search. *SIAM J. Comput.* **43**(2), 514–542 (2014)
11. Fischer, M., Nemhauser, G., Wolsey, L.: An analysis of approximations for maximizing submodular set functions-II. *Math. Program. Stud.* **8**, 73–87 (1978)
12. Goldenberg, J., Libai, B., Muller, E.: Talk of the network: a complex systems look at the underlying process of word-of-mouth. *Mark. Lett.* **12**(3), 211–223 (2001)
13. Iyer, R.K., Bilmes, J.A.: Submodular optimization with submodular cover and submodular knapsack constraints. In: *Advances in Neural Information Processing Systems*, pp. 2436–2444 (2013)
14. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 137–146 (2003)
15. Kim, G., Xing, E.P., Fei-Fei, L., Kanade, T.: Distributed cosegmentation via submodular optimization on anisotropic diffusion. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 169–176 (2011)
16. Krause, A., Golovin, D.: *Tractability: Practical Approaches to Hard Problems*. Submodular function maximization, vol. 3. Cambridge University Press, Cambridge (2012)
17. Krause, A., Singh, A., Guestrin, C.: Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.* **9**, 235–284 (2008)
18. Krause, A., McMahan, B., Guestrin, C., Gupta, A.: Selecting observations against adversarial objectives. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 777–784 (2008)
19. Lee, J., Sviridenko, M., Vondrák, J.: Submodular maximization over multiple matroids via generalized exchange properties. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 244–257 (2009)
20. Lin, H., Bilmes, J.: Multi-document summarization via budgeted maximization of submodular functions. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 912–920
21. Lin, H., Bilmes, J.: A class of submodular functions for document summarization. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 510–520 (2011)
22. Nemhauser, G.L., Wolsey, L.A.: Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.* **3**(3), 177–188 (1978)

23. Nemhauser, G., Wolsey, L., Fisher, M.: An analysis of approximations for maximizing submodular set functions-I. *Math. Program.* **14**(1), 265–294 (1978)
24. Oxley, J.: *Matroid Theory*. Oxford University Press, Oxford (1992)
25. Streeter, M., Golovin, D.: An online algorithm for maximizing submodular functions. Carnegie Mellon University, Technical report (2007)
26. Streeter, M., Golovin, D.: An online algorithm for maximizing submodular functions. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 1577–1584 (2008)
27. Vondrák, J., Chekuri, C., Zenklusen, R.: Submodular function maximization via the multilinear relaxation and contention resolution schemes. In: *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, pp. 783–792 (2011)
28. Wolsey, L.: An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica* **2**(4), 385–393 (1982)
29. Zheng, Z., Shroff, N.B.: Submodular utility maximization for deadline constrained data collection in sensor networks. *IEEE Trans. Autom. Control* **59**(9), 2400–2412 (2014)

Submodularity in Dynamics and Control of Networked
Systems

Clark, A.; Alomair, B.; Bushnell, L.; Poovendran, R.

2016, XVII, 210 p. 63 illus., 15 illus. in color., Hardcover

ISBN: 978-3-319-26975-7