Maximization of Non-Monotone Submodular Functions

Jennifer Gillenwater

Abstract

A litany of questions from a wide variety of scientific disciplines can be cast as non-monotone submodular maximization problems. Since this class of problems includes max-cut, it is NP-hard. Thus, general-purpose algorithms for the class tend to be approximation algorithms. For unconstrained problem instances, one recent innovation in this vein includes an algorithm of Buchbinder et al. (2012) that guarantees a $\frac{1}{2}$-approximation to the maximum. Building on this, for problems subject to cardinality constraints, Buchbinder et al. (2014) offer guarantees in the range $[0.356, \frac{1}{2} + o(1)]$. Earlier work has the best approximation factors for more complex constraints and settings. For constraints that can be characterized as a solvable polytope, Chekuri et al. (2011) provide guarantees. For the online secretary setting, Gupta et al. (2010) provide guarantees. In sum, the current body of work on non-monotone submodular maximization lays strong foundations. However, there remains ample room for future algorithm development.

Contents

List of Algorithms

Chapter 1

Introduction

This survey focuses on recent developments in the maximization of non-monotone submodular functions. Specifically, analysis of the following four publications will constitute the core of the review:

- Buchbinder et al. (2012): provides the best guarantee for the unconstrained setting.
- Buchbinder et al. (2014): provides the best guarantee for the cardinality-constrained setting.
- Chekuri et al. (2011): provides guarantees for the setting where the convex hull of the feasible sets forms a down-monotone, solvable polytope.
- Gupta et al. (2010): provides guarantees for the online setting.

Before going into the specifics of each paper, we begin with some introductory material. Section 1.1 motivates our topic of study by illustrating the pervasiveness and practical importance of submodular maximization. Section 1.2 provides technical background material on submodularity, non-monotonicity, and common constraints for submodular maximization problems. Section 1.3 discusses in more detail the example problems from Section 1.1.

1.1 Motivation

Submodular maximization subsumes a wide variety of problems. It encompasses some of the very general NP-hard problems taught in introductory algorithms courses, such as max-cut, max-dicut, max-k-coverage, and facility location. (See Frieze (1974) and Chapter 6 of Bach (2013) for proofs of their submodularity.) It also contains variations on max-SAT and max-welfare (see Section 4 of Buchbinder et al. (2012)). More concretely, a multitude of real-world tasks can be formulated as submodular maximization problems. A few examples are listed here, along with references to papers containing more detailed descriptions of each task's nuances. Section 1.3 also provides a more precise description of these tasks.

- Document summarization (Lin and Bilmes, 2012; Kulesza and Taskar, 2012)
- Sensor placement (Krause and Guestrin, 2005; Krause et al., 2008)
- Social network marketing (Hartline et al., 2008; Mossel and Roch, 2007)
- Cooperative game design (Schulz and Uhan, 2013)
- Image segmentation (Kim et al., 2011)
- Auction revenue maximization (Dughmi et al., 2009)

Note that some of the attempts to tackle these tasks rely on *monotone* rather than *non-monotone* submodular functions. For example, in the realm of document summarization, the work of Lin and Bilmes (2012) uses monotone functions. However, empirical results (see Table 4 of Kulesza and Taskar (2012)) indicate that a similar but *non-monotone* function seems to be a better fit for the problem. This is one of the reasons our survey focuses on the non-monotone realm—non-monotone functions can sometimes offer a more realistic representation of the underlying mechanics of real-world systems.

1.2 Technical Background

Consider a ground set consisting of N items: $\Omega = \{1, \ldots, N\}$. A set function $f : 2^\Omega \to \mathbb{R}$ maps each subset of this ground set Ω to a real number.

1.2.1 Submodularity

A set function qualifies as *submodular* if it obeys the law of diminishing returns.

Definition 1. (Submodularity by diminishing returns) *The function $f : 2^\Omega \to \mathbb{R}$ is submodular if and only if $\forall A, B \subseteq \Omega$ such that $A \subseteq B$ and $i \in \Omega \setminus B$, it is the case that $f(B \cup \{i\}) - f(B) \leq f(A \cup \{i\}) - f(A)$.*

In words: a new element contributes more value when added to set A than when added to any superset of A. Diminishing returns is equivalent to several other simple conditions.

Definition 2. (Submodularity by set combination) *The function $f : 2^\Omega \to \mathbb{R}$ is submodular if and only if $\forall A, B \subseteq \Omega$ it is the case that $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$.*

Definition 3. (Submodularity by second-order differences) *The function $f : 2^\Omega \to \mathbb{R}$ is submodular if and only if $\forall A \subseteq \Omega$ and $i, j \in \Omega \setminus A$, it is the case that $f(A \cup \{i, j\}) - f(A \cup \{j\}) \leq f(A \cup \{i\}) - f(A)$.*

Note that while Definition 3 seems weaker than Definition 1, as it corresponds to restricting to $B = A \cup \{j\}$, it is nonetheless equivalent. (See Propositions 2.2 and 2.3 of Bach (2013) for proofs of these definitions' equivalence.) Given a submodular function, many simple variations on it are also submodular. For example, we show here that the complement of a submodular function is also submodular.

Theorem 4. *For submodular f, its complement $\bar{f}(S) = f(\Omega \setminus S)$ is also submodular.*

Proof. Using sets $A \subseteq B \subseteq \Omega$ and element $i \in \Omega \setminus B$ (as in Definition 1), we have the following equivalence.

$$\begin{aligned} \bar{f}(B \cup \{i\}) - \bar{f}(B) &= f(\Omega \setminus (B \cup \{i\})) - f(\Omega \setminus B) && (1.1) \\ &= f(\Omega \setminus (B \cup \{i\})) - f(\Omega \setminus (B \cup \{i\}) \cup \{i\}) && (1.2) \\ &= f(C) - f(C \cup \{i\}) && (1.3) \end{aligned}$$

where C is defined as $\Omega \setminus (B \cup \{i\})$. Similarly, define $D = \Omega \setminus (A \cup \{i\})$. Then $A \subseteq B$ implies $C \subseteq D$. Since $i \notin D$, Definition 1 of submodularity can be written in terms of C, D, and i: $f(D \cup \{i\}) - f(D) \leq f(C \cup \{i\}) - f(C)$. Applying Equation (1.3) and the analogous equality for A and D, this inequality becomes: $\bar{f}(B \cup \{i\}) - \bar{f}(B) \leq \bar{f}(A \cup \{i\}) - \bar{f}(A)$. □

1.2.2 NP-Hardness and Monotonicity

The following optimization problem is NP-hard, since it generalizes max-cut.

Problem 5. (Unconstrained submodular maximization) *For a submodular function $f : 2^\Omega \to \mathbb{R}_+$, find:* $\max_{S \subseteq \Omega} f(S)$.

The max-cut problem is significantly harder than many other submodular problems though. In fact, max-cut and many of the hardest submodular problems can be characterized as *non-monotone*.

Definition 6. (Monotonicity) *The submodular function $f : 2^\Omega \to \mathbb{R}$ is monotone if and only if $\forall A, B \subseteq \Omega$, such that $A \subseteq B$, it is the case that $f(A) \leq f(B)$.*

In words: adding elements can never decrease the value of a monotone submodular function. The monotonic property can be used to divide the class of submodular functions. Even restricting to monotone f though, the maximization problem is not always easy. The addition of the simple cardinality constraint $|S| \leq k$ makes the problem NP-hard, since it generalizes max-k-cover. Due to the maximization problem's hardness, the arena of submodular maximization algorithms is populated by approximation algorithms. For example, one commonly-used algorithm is that of Nemhauser et al. (1978), shown as Algorithm 1 here. Essentially, the algorithm starts from the empty set and greedily adds one item every iteration until no additions can increase the score.

Algorithm 1 NEMHAUSER-WOLSEY

Input: submodular function f, ground set Ω, limit k
$S \leftarrow \emptyset, U \leftarrow \Omega$
while $|S| < k$ **do**
  $i^* \leftarrow \arg\max_{i \in U} f(S \cup \{i\})$
  if $f(S \cup \{i^*\}) < f(S)$ **then**
    break
  end if
  $S \leftarrow S \cup \{i^*\}$
  $U \leftarrow U \setminus \{i^*\}$
end while
return S

As with *all* of the algorithms we will consider, this one only provides an approximation guarantee when f, in addition to being submodular, is non-negative.

Definition 7. (Non-negative) *The set function $f : 2^\Omega \to \mathbb{R}$ is non-negative if and only if $\forall S \subseteq \Omega$, $f(S) \geq 0$.*

Without this property, an exponential number of calls to an f-oracle is needed to even decide whether the maximum value is positive or zero (see Footnote 4 in Chekuri et al. (2011)). Notice that we already included the non-negativity property in the definition of Problem 5 by making f's range $\mathbb{R}_+$. One additional property, normalization, is necessary before stating the approximation guarantee for the NEMHAUSER-WOLSEY algorithm.

Definition 8. (Normalization) *The set function $f : 2^\Omega \to \mathbb{R}$ is normalized if and only if $f(\emptyset) = 0$.*

Nemhauser et al. (1978)'s algorithm provides a $(1 - 1/e)$-approximation guarantee for *normalized, non-negative, monotone* submodular functions. There is no guarantee, however, when using this algorithm with a *non-monotone* function, which motivates the work we will describe in Chapters 2 and 3.

1.2.3 Independence Systems

In addition to Problem 5, constrained variants of submodular maximization are also of practical interest. Besides simple cardinality constraints, $|S| = k$ or $|S| \leq k$, constraints based on independence systems and matroids are common. Independence systems are a generalization of the notion of linear independence that applies to vectors.

Definition 9. (Independence system) *A (finite) independence system is a pair $(\Omega, \mathcal{I})$ where Ω is a finite set and $\mathcal{I}$ is a family of subsets of Ω. The family $\mathcal{I}$ must be such that:*

- $\emptyset \in \mathcal{I}$, *and*
- $\mathcal{I}$ *has the "hereditary" property:* $I_1 \subseteq I_2 \in \mathcal{I}$ *implies* $I_1 \in \mathcal{I}$.

The sets in $\mathcal{I}$ are called the system's "independent sets". (All other subsets of Ω are called dependent sets.) It is often useful to constrain a submodular maximization problem to the independent sets of some independence system. A maximal independent set — a set that becomes dependent with the addition of any item — is called a "basis" (of Ω). The size of the largest basis is the independence system's "rank". The notion of bases and rank extend to all $S \subseteq \Omega$.

Definition 10. (Independence system basis) *For any given set* $S \subseteq \Omega$, *a set* $S' \subseteq S$ *is a maximal independent set of* S *if the addition of any element* $i \in S \setminus S'$ *to* S' *would make* $S' \cup \{i\}$ *dependent. These maximal independent sets of* S, *which we will denote* $\mathcal{B}(S)$, *are called the bases of* S.

Definition 11. (Independence system rank) *For any given set* $S \subseteq \Omega$, *its rank is the size of its largest independent set. The rank function,* $r : 2^\Omega \to \mathbb{Z}_+$, *of an independence system, maps sets to their ranks:* $r(S) = \max_{S' \in \mathcal{B}(S)} |S'|$.

Independence systems are sometimes classified according to the ratio of their maximum and minimum basis sizes.

Definition 12. (p-independence system) *Let the function* $\eta : 2^\Omega \to \mathbb{Z}_+$ *map sets to the size of their smallest basis:* $\eta(S) = \min_{S' \in \mathcal{B}(S)} |S'|$. *Then an independence system* $(\Omega, \mathcal{I})$ *qualifies as a* p*-system if* $\max_{S \subseteq \Omega} \frac{r(S)}{\eta(S)} \leq p$.

A related but more restrictive class of constraints are provided by matroids. Matroids are a sub-class of independence systems. They require that $\mathcal{I}$ have one additional property.

Definition 13. (Matroid) *A matroid is an independence system, with the additional requirement that* $\mathcal{I}$ *has the "exchange" property:* $\forall I_1, I_2 \in \mathcal{I}$, $|I_1| < |I_2|$ *implies* $\exists i \in I_2 \setminus I_1$ *such that* $I_1 \cup \{i\} \in \mathcal{I}$.

Every matroid qualifies as a 1-independence system. The intersection of p matroids, while not necessarily a matroid, qualifies as a p-independence system. A few examples of commonly-used matroids are given below.

- **Uniform matroid**: Let $\mathcal{I}$ be the family of all $S \subseteq \Omega$ that are at most size k: $|S| \leq k$. Then $\mathcal{M} = (\Omega, \mathcal{I})$ is a matroid. Its bases are the sets of size exactly k.
- **Linear matroid**: Given a matrix M with N columns, let $\Omega = \{1, \ldots, N\}$ index the columns. Let $\mathcal{I}$ be the family of all sets of column indices whose corresponding columns are linearly independent. Then $\mathcal{M} = (\Omega, \mathcal{I})$ is a matroid. The rank of $\mathcal{M}$ is equal to the rank of the matrix M.
- **Graphic matroid**: Given a graph $G = (V, E)$ with vertices V and edges E, let $\mathcal{I}$ be the family of all edge sets that do not contain a cycle. Then $\mathcal{M} = (E, \mathcal{I})$ is a matroid. Assuming G is a single connected component, the bases of $\mathcal{M}$ are the spanning trees of G. If G has multiple connected components, the bases of $\mathcal{M}$ are the forests of G.
- **Matching matroid**: Given a graph $G = (V, E)$, let $\mathcal{I}$ be the family of all $S \subseteq V$ that can be covered by a matching. (A set of edges is a matching if no two edges share a vertex.) Then $\mathcal{M} = (E, \mathcal{I})$ is a matroid. The bases of $\mathcal{M}$ correspond to subsets of V that are maximum matchings in G.
- **Partition matroid**: Let $P_1, \ldots, P_k$ be a partition of Ω. Let $a_1, \ldots, a_k$ be constants associated with the partitions. Finally, let $\mathcal{I}$ be the family of all sets I that satisfy $|I \cap P_i| \leq a_i \ \ \forall i \in \{1, \ldots, k\}$. Then $\mathcal{M} = (\Omega, \mathcal{I})$ is a matroid.

To conclude this section and give a preview of the next section, we consider two concrete problems. The first one shows how a simple, modular function subject to a matroid constraint can describe an interesting practical problem.

Problem 14. (Max-vertex-matching): *Suppose that every element i in the ground set of a matching matroid $\mathcal{M} = (\Omega, \mathcal{I})$ has an associated weight, w_i. Then $\max_{S \in \mathcal{I}} \sum_{i \in S} w_i$ corresponds to finding the maximum-weight vertex set that constitutes a matching.*

A related problem constitutes our second example, and illustrates the practicality of independence systems. Note that the intersection of two (or more) matroids, $\mathcal{M}_1 \cap \mathcal{M}_2 = (\Omega, \mathcal{I}_1 \cap \mathcal{I}_2)$, is not necessarily itself a matroid. However, it is an independence system. Thus, when working with problems subject to multiple matroid constraints, it is practical to apply algorithms designed to handle independence systems.

Problem 15. (Max-bipartite-edge-matching): *Let $G = (U, V, E)$ represent a bipartite graph. Define a partition matroid $\mathcal{M}_U = (E, \mathcal{I}_U)$ with $|U|$ parts to the partition: for node u, define $P_u = \{(a, b) \in E \mid a = u \vee b = u\}$, the edges for which vertex u is an endpoint. Set each intersection limit at $a_u = 1$. Then each independent set $I \in \mathcal{I}_U$ is a set of edges of which no two have the same endpoint in U. Let $\mathcal{M}_V$ be the analogous partition matroid for V. Then any independent set of the independence system $\mathcal{M}_U \cap \mathcal{M}_V$ consists of the* edges *of a bipartite matching. Associating a weight w_e with each edge e, the maximum-weight edge set that constitutes a matching is: $\max_{S \in \mathcal{I}_U \cap \mathcal{I}_V} \sum_{e \in S} w_e$.*

1.3 Example Problems

Having covered the necessary background material, we return now to the example problems listed in Section 1.1 to discuss them in more detail.

1.3.1 General NP-Hard Problems

First, recall that very general NP-hard problems such as max-cut, max-dicut, max-k-coverage, facility location, and variations on max-SAT and max-welfare are submodular. Descriptions of submodular variants of max-SAT and max-welfare we defer to Chapter 2, but the other four problems we define here.

Problem 16. (Max-cut): *Given a graph $G = (V, E)$, let $g : 2^V \to 2^E$ be a function from subsets of V to subsets of E such that $g(S) = \{$edges from S to $V \setminus S\}$. Then, find: $\max_{S \subseteq V} |g(S)|$.*

Max-dicut is identical to max-cut, but for a graph with directed edges. For max-cut and max-dicut there exist 0.878- and 0.796-approximation algorithms (Goemans and Williamson, 1995). Unfortunately though, these algorithms do not easily extend to all submodular functions. Max-k-coverage, on the other hand, is best approximated, with a factor of $(1 - \frac{1}{e})$ (Khuller et al., 1999), by the general submodular optimization algorithm we saw in the previous section (Algorithm 1).

Problem 17. (Max-k-coverage): *Given sets $\{A_1, \ldots, A_N\}$, find: $\max_{S : |S| \leq k} \left| \bigcup_{i \in S} A_i \right|$.*

Facility location, like the cut problems, has a specialized algorithm. It comes with a 0.828-approximation guarantee (Ageev and Sviridenko, 1999), but again, isn't easily generalized to all submodular functions.

Problem 18. (Facility location): *Given two sets, $\Omega = \{1, \ldots, N\}$ (possible facility locations) and $J = \{1, \ldots, M\}$ (clients), and non-negative rational numbers $p_{ij}, c_i \;\; \forall i \in \Omega, \; j \in J$, find:*

$$\max_{S \subseteq \Omega} \sum_{j \in J} \max_{i \in S} p_{ij} - \sum_{i \in S} c_i \; . \tag{1.4}$$

The p_{ij} are usually interpreted as profits from serving client j with facility i, while c_i is the cost of opening a facility at location i.

1.3.2 Real-World Tasks

We now delve into the more application-based tasks listed in Section 1.1. Note that ultimately some of these amount to variations on the general NP-hard problems from the previous subsection.

- **Document summarization (Lin and Bilmes, 2012; Kulesza and Taskar, 2012)**: Given a set of documents consisting of a total of N sentences, the goal is to select a subset S of the sentences that summarizes the core content of the documents. Usually, this is accompanied by a constraint on the summary size, such as $|S| \leq k$. Common submodular functions used in this case are those that encourage the sentences to cover as much of the material as possible. This includes the monotone submodular Problem 17 (max-k-coverage), as well as the non-monotone max-log-det:

  Problem 19. (Max-log-det) *Given a positive semi-definite matrix L, find:*

  $$\max_S \log\det(L_S) \tag{1.5}$$

  where the subscript notation L_S indicates the sub-matrix consisting of the rows and columns indexed by S.

  For the document summarization task, one reasonable way to form the matrix L is through dot products of feature vectors. Specifically, if sentence i is characterized by length-M feature vector b_i, let B represent the $N \times M$ matrix where each feature vector is a row. Then for $L = BB^\top$, Problem 19 corresponds to finding a diverse set of sentences.

- **Sensor placement for Gaussian Processes (Krause et al., 2008)**: Suppose there are N possible locations for sensors (e.g. for measuring pressure, temperature, or pollution), and an $N \times N$ covariance matrix Σ estimating the correlation of measurements at the various locations. The goal is to select a subset S of locations at which to actually place sensors, such that the measurements from these sensors will be as informative as possible about the space as a whole. Common submodular problems that fit this task well include max-entropy and max-mutual-information:

  Problem 20. (Max-entropy) *Given a set of random variables $\{X_1, \ldots, X_N\}$, find:*

  $$\max_{S:|S|\leq k} H(\{X_i : i \in S\}) \,. \tag{1.6}$$

  Note that this is equivalent to minimizing the following conditional entropy:

  $$H(\{X_i : i \notin S\} \mid \{X_i : i \in S\}) = H(\{X_1, \ldots, X_N\}) - H(\{X_i : i \in S\}) \,. \tag{1.7}$$

  Problem 21. (Max-mutual-information) *Given a set of random variables $\{X_1, \ldots, X_N\}$, find:*

  $$\max_{S:|S|\leq k} I(\{X_i : i \in S\}; \{X_i : i \notin S\}) = \max_{S:|S|\leq k} H(\{X_i : i \notin S\}) - H(\{X_i : i \notin S\} \mid \{X_i : i \in S\}) \,. \tag{1.8}$$

- **Sensor placement with Graphical Models (Krause and Guestrin, 2005)**: As for the previous sensor placement case, the input here consists of N possible locations for sensors. Instead of assuming a measurement covariance matrix though, in this setting it is assumed that the sensors are nodes in a graphical model. The same submodular objective functions as before, such as Problem 21, can be used to select a subset S of the nodes. Computing the objective value is efficient as long as the structure of the graphical model is such that standard inference in it is efficient.

- **Social network marketing (Hartline et al., 2008)**: Consider a social network consisting of N people, with weights w_{ij} indicating the influence that person i has on person j. Suppose a seller has a digital good that costs nothing to copy (unlimited supply), and they would like to maximize the revenue r from selling this good within the network. In this setting, a common marketing strategy is "influence-and-exploit": give the product to a subset S of the network for free, then fix a price at which to offer it to the rest of the network. The selection of the initial subset S is often formulated as a submodular maximization problem. More precisely, let the strength of the relationship between person i and person j be w_{ij}. Then it is assumed that the value person i places on the product is: $v_i(S) = g_i(\sum_{j \in S \cup \{i\}} w_{ij})$, where g_i is any non-negative, monotone, concave function. Finding $\max_{S:|S| \leq k} \sum_{i=1}^{N} v_i(S)$ is one way to select the initial set of people who will get the product for free.

- **Cooperative game design (Schulz and Uhan, 2013)**: Consider a game with N players, where it would be ideal to have all players cooperate (form a single coalition). For example, players might be corporations that all need to dispose of toxic byproducts from their manufacturing plants. The government might want to encourage the corporations to pool the waste at a single site. Thus, it needs to answer the question, "How much should opening a new waste site cost?". In other words, "How much of a penalty is necessary to get all players to cooperate?". Suppose the supermodular function $g(S)$ represents the total cost to players in S if they cooperate. Let x_i represent the cost to player i when all players cooperate. Then the following submodular maximization problem identifies the coalition whose dissatisfaction with the single-coalition outcome is largest: $\max_S \sum_{i \in S} x_i - g(S)$. The government must then apply a penalty that is sufficient to induce the most dissatisfied coalition to join forces with the other players.

- **Image segmentation (Kim et al., 2011)**: The initial step in most image-manipulation software is to segment a given image into k parts such that it is homogeneous within each part (e.g. separating an image into "sky" and "earth" is a common task, with $k = 2$). Typically, image segmentation is thought of as a submodular *minimization* problem, a variant of min-cut. However, an alternative specification as a submodular *maximization* problem is also possible. Consider formulating the segmentation task as the problem of identifying k pixels (or superpixels) to serve as "centers" for the segments. The similarity between a pair of adjacent pixels, $\text{sim}(i, j)$, can be thought of as the probability that "heat" applied to one pixel will diffuse to the other. In other words, the image segmentation problem is analogous to the problem of selecting k heat sources in order to maximize the temperature of a system under heat diffusion. If the diffusion is linear and anisotropic, then the overall temperature of the system turns out to be a submodular function of the k sources. Specifically, let $\Omega = \{1, \ldots, N\}$ represent the set of all pixels (including an auxiliary set of pixels added to act as the border of the image). Let $\mathcal{N}(i)$ be the neighbors of pixel i. Then the "temperature" of pixel i is:

$$t(i) = \begin{cases} 1 & \text{if } i \text{ is selected as one of the } k \text{ heat sources} \\ 0 & \text{if } i \text{ is on the image border} \\ \sum_{j \in \mathcal{N}(i)} \text{sim}(i, j) t(j) & \text{otherwise.} \end{cases} \tag{1.9}$$

  The overall temperature of the system is: $\sum_{i \in \Omega} t(i)$. The k centers needed to segment the image can be found by maximizing this submodular expression.

- **Auction revenue maximization (Dughmi et al., 2009)**: Given a base set of bidders B, a pool of potential bidders Ω, and a budget k, a common task is to recruit a subset $S \subseteq \Omega$ of k additional bidders for the auction. This is sometimes called the "market expansion problem". The objective to maximize in this case is auction revenue. It can be shown that if the standard Vickrey-Clarke-Groves (VCG) auction mechanism is applied, the resulting revenue function is submodular, as long as a few additional conditions apply. (These conditions concern the size of the known bidders B and the bidders' valuations for the auction items; see Theorem 3.2 of Dughmi et al. (2009) for details.)

In some of the above applications, especially where the objective function is monotone, cardinality constraints are present. Knapsack constraints are clearly often practical as well, whenever the items concerned have an associated cost and the task has a budget. The practicality of independence system constraints is perhaps harder to see, so we finish this section by giving an example use.

- **Image segmentation subject to an independence system constraint**: Recall the image segmentation task. If there are many images to segment into earth/sky parts, it makes sense to perform joint inference; intuitively, the location of the sky's center in one image is likely to be close to its location in the other images. To make use of this intuition, it would be convenient to add a matching term to the objective, matching sky in image 1 to sky in image 2 and penalizing by how much their locations differ. This is a straightforward application of Problem 15 (max-bipartite-edge-matching). For a pair of images, $i = \{1, 2\}$, with the segment centers (c_1^i, c_2^i), define a fully connected bipartite graph with vertices $U = \{c_1^1, c_2^1\}$ and $V = \{c_1^2, c_2^2\}$. Let the weight of an edge correspond to how similar the locations of its endpoints are. Add this (modular) Problem 15 objective to the original, submodular image segmentation objective. The result is a practical, submodular optimization problem, constrained by an independence system.

The remaining sections of this survey will primarily focus on *non-monotone* submodular functions. We should pause here and briefly note that many of the applications from this section clearly rely on *monotone* functions. However, the non-monotone advancements may be helpful for them as well, for two reasons:

- First, as mentioned in Section 1.1, the non-monotone max-log-det (Problem 19) formulation has been shown to work better in practice for document summarization than some monotone submodular functions (Kulesza and Taskar, 2012). This insight may carry over into other problems.
- Secondly, in practice the more complex algorithms developed for non-monotone maximization may work well for monotone maximization too — not necessarily in general, but perhaps for a particular problem instance or distribution over instances.

Chapter 2

Unconstrained Setting

We begin by discussing the simplest setting, that with no constraints on the selected set S. This is exactly what was defined in the previous chapter as Problem 5. Accordingly, the algorithms in this chapter bear the suffix "USM" for "unconstrained submodular maximization". Feige et al. (2007) prove that for this setting attaining better than a $\frac{1}{2}$-approximation is hard in the value oracle sense. That is, according to Theorem 4.5 of their work, it would require an exponential number of f-value queries, $e^{\varepsilon^2 N/16}$, for any $\varepsilon > 0$.

Proof sketch: Feige et al. (2007) construct an f such that for most sets its value is that of the cut function of a complete graph with edge weights of 1 (max value: $\frac{1}{4}N^2$). But for a few sets f has the value of a cut function on a complete bipartite graph with edge weights of 2 (max value: $\frac{1}{2}N^2$). They show that any algorithm would have to query f an exponential number of times to be assured of finding the bipartite portion of f.

More recently, Dobzinski and Vondrák (2012) translate this result from the value oracle context to complexity theory. Theorem 3.1 of their work states that better than a $\frac{1}{2}$-approximation implies NP = RP.

Proof sketch: Using the same f as Feige et al. (2007), Dobzinski and Vondrák (2012) show how to represent f compactly and explicitly such that distinguishing between the two types of graphs implies deciding Unique-SAT. Unique-SAT is the problem of deciding whether a Boolean formula has exactly one satisfying assignment. There is a randomized polynomial-time (RP) reduction from SAT to Unique-SAT (see Theorem 1.1 of Valiant and Vazirani (1986)), so Unique-SAT cannot be solved in polynomial time unless NP = RP.

2.1 Selected Algorithms

In light of these hardness results, the best that can be hoped for is a $\frac{1}{2}$-approximation. Buchbinder et al. (2012), whose work we will focus on for the remainder of this chapter, present a surprisingly simple algorithm, shown as Algorithm 3 here, that achieves exactly this. A deterministic version, Algorithm 2, yields a $\frac{1}{3}$-approximation. The two algorithms are very similar, with the main difference being that for Algorithm 3 the decision of whether or not to incude an item is softened. Note that in that algorithm, if $a'_i = b'_i = 0$, then it is assumed that the item i gets included.

2.2 Selected Proofs

The proofs of the approximation guarantees for both of these algorithms are very similar, so here we will focus on the $\frac{1}{2}$-approximation proof. It relies on two supporting lemmas. The first lemma says that the change in f upon adding item i to X_{i-1} can only be as small as the negative of the change observed when removing i from Y_{i-1}. Intuitively, this means that one of the two possible actions, adding or removing, has a relatively positive impact on the value of f.

Algorithm 2 DETERMINISTIC-USM

Input: submodular function f, ground set Ω
$X_0 \leftarrow \emptyset, Y_0 \leftarrow \Omega$
for $i = 1$ **to** N **do**
  $a_i \leftarrow f(X_{i-1} \cup \{i\}) - f(X_{i-1})$
  $b_i \leftarrow f(Y_{i-1} \setminus \{i\}) - f(Y_{i-1})$
  if $a_i \geq b_i$ **then**
    $X_i \leftarrow X_{i-1} \cup \{i\}, Y_i \leftarrow Y_{i-1}$
  else
    $X_i \leftarrow X_{i-1}, Y_i \leftarrow Y_{i-1} \setminus \{i\}$
  end if
end for
return X_n

Algorithm 3 RANDOMIZED-USM

Input: submodular function f, ground set Ω
$X_0 \leftarrow \emptyset, Y_0 \leftarrow \Omega$
for $i = 1$ **to** N **do**
  $a_i \leftarrow f(X_{i-1} \cup \{i\}) - f(X_{i-1})$
  $b_i \leftarrow f(Y_{i-1} \setminus \{i\}) - f(Y_{i-1})$
  $a'_i \leftarrow \max\{a_i, 0\}, b'_i \leftarrow \max\{b_i, 0\}$
  with probability $\frac{a'_i}{a'_i + b'_i}$ **do:**
    $X_i \leftarrow X_{i-1} \cup \{i\}, Y_i \leftarrow Y_{i-1}$
  else do:
    $X_i \leftarrow X_{i-1}, Y_i \leftarrow Y_{i-1} \setminus \{i\}$
end for
return X_n

Lemma 22. *For each $i \in \{1, \ldots, N\}$, $a_i + b_i \geq 0$.*

Proof. Let $A = X_{i-1} \cup \{u_i\}$ and $B = Y_i \setminus \{u_i\}$. Then $A \cup B = \{u_i\} \cup Y_i = Y_{i-1}$ and $A \cap B = X_{i-1}$. Applying Definition 2 of submodularity, we have:

$$f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}) + f(Y_i \setminus \{u_i\}) - f(Y_{i-1}) \geq 0 \,. \tag{2.1}$$

The lefthand side of this equation is exactly $a_i + b_i$, which proves the sum is always non-negative. □

The second lemma establishes that a set OPT_i, whose composition depends on the true OPT and the algorithm's X_i, does not decrease too much in value from one iteration to the next. Formally, OPT_i is: $\mathrm{OPT}_i = (\mathrm{OPT} \cup X_i) \cap Y_i$. At the beginning of the algorithm $\mathrm{OPT}_0 = \mathrm{OPT}$. At iteration i, OPT_i equals X_i for items $1, \ldots, i$, but still equals OPT for all items not yet considered (items $i+1, \ldots, N$). At the end of the algorithm, $\mathrm{OPT}_N = X_N$, the set returned by Algorithm 3. It will be useful in the proof of the second lemma to further observe that:

Observation 23. *If $i \notin \mathrm{OPT}$, then $\mathrm{OPT}_{i-1} \subseteq Y_{i-1} \setminus \{i\}$.*

Observation 24. *If $i \in \mathrm{OPT}$, then $X_{i-1} \subseteq \mathrm{OPT}_{i-1} \setminus \{i\}$.*

We now prove the final supporting lemma.

Lemma 25. *Define $Z_i \sim$ Uniform$(0, 1)$, and let $Z_i \leq \frac{a_i}{a_i + b_i}$ result in Algorithm 3 adding i to X_{i-1}. Then, for each $i \in \{1, \ldots, N\}$:*

$$\mathbb{E}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)] \leq \frac{1}{2}\mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] \tag{2.2}$$

where the expectations are with respect to the Z_i.

Proof. At step i, the algorithm has already fixed some set $X_{i-1} \subseteq \{1, \ldots, i-1\}$. Any such set also determines a_i and b_i. Thus, we can break down the analysis by the possible value combinations of a_i and b_i.

- $\mathbf{a_i \geq 0}$ and $\mathbf{b_i \leq 0}$: Item i is always included in X_i in this case. Thus, $f(Y_i) - f(Y_{i-1}) = 0$ and we only have to show that $f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{i\}) \leq \frac{a_i}{2}$. If $i \in \mathrm{OPT}$, we are already done, since $0 \leq \frac{a_i}{2}$. Otherwise, let $A = \mathrm{OPT}_{i-1}$ and let $B = Y_{i-1} \setminus \{i\}$. Recall that Observation 23 establishes $A \subseteq B$. Then, applying Definition 1 of submodularity, we have:

  $$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{i\}) \leq f(Y_{i-1} \setminus \{i\}) - f(Y_{i-1}) = b_i \,. \tag{2.3}$$

  Since $b_i \leq 0 \leq \frac{a_i}{2}$, we are again done.

- $\mathbf{a_i < 0}$ and $\mathbf{b_i \geq 0}$: This case is analogous to the previous one. Now, item i is never included in X_i, which makes $f(X_i) - f(X_{i-1}) = 0$. Then we only have to show that $f(\text{OPT}_{i-1}) - f(\text{OPT}_{i-1} \setminus \{i\}) \leq \frac{b_i}{2}$. If $i \notin \text{OPT}$, we are already done, since $0 \leq \frac{b_i}{2}$. Otherwise, let $A = X_{i-1}$ and let $B = \text{OPT}_{i-1} \setminus \{i\}$. Recall that Observation 24 establishes $A \subseteq B$. Then, applying Definition 1 of submodularity, we have:

$$f(\text{OPT}_{i-1}) - f(\text{OPT}_{i-1} \setminus \{i\}) \leq f(X_{i-1} \cup \{i\}) - f(X_{i-1}) = a_i \,. \tag{2.4}$$

  Since $a_i < 0 \leq \frac{b_i}{2}$, we are again done.

- $\mathbf{a_i \geq 0}$ and $\mathbf{b_i > 0}$: With probability $\frac{a_i}{a_i + b_i}$ the code branch with f-value change a_i is taken, and with probability $\frac{b_i}{a_i + b_i}$ the code branch with f-value change b_i is taken. Thus, we have:

$$\mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] = \frac{a_i}{a_i + b_i} \cdot a_i + \frac{b_i}{a_i + b_i} \cdot b_i = \frac{a_i^2 + b_i^2}{a_i + b_i} \,. \tag{2.5}$$

  Similarly, breaking down the cases for the other side of the inequality that we are trying to prove yields:

$$\begin{aligned} \mathbb{E}[f(\text{OPT}_{i-1}) - f(\text{OPT}_i)] = {} & \frac{a_i}{a_i + b_i} [f(\text{OPT}_{i-1}) - f(\text{OPT}_{i-1} \cup \{i\})] + \\ & \frac{b_i}{a_i + b_i} [f(\text{OPT}_{i-1}) - f(\text{OPT}_{i-1} \setminus \{i\})] \,. \end{aligned} \tag{2.6}$$

  This can be further simplified by analyzing the two cases of $i \in \text{OPT}$ and $i \notin \text{OPT}$:

  - $\mathbf{i} \in \text{OPT}$: The first term in Equation (2.6) is zero. The second term is bounded above by b_i, using Observation 24 and applying Definition 1 of submodularity as before.
  - $\mathbf{i} \notin \text{OPT}$: The second term in Equation (2.6) is zero. The first term is bounded above by a_i, using Observation 23 and applying Definition 1 of submodularity as before.

  Thus, $\mathbb{E}[f(\text{OPT}_{i-1}) - f(\text{OPT}_i)] \leq \frac{a_i b_i}{a_i + b_i}$. It remains to show that:

$$\frac{a_i b_i}{a_i + b_i} \leq \frac{1}{2} \left(\frac{a_i^2 + b_i^2}{a_i + b_i} \right) \,. \tag{2.7}$$

  Since the common denominator $a_i + b_i \geq 0$, the above inequality is true so long as the numerator difference $\frac{1}{2}(a_i^2 + b_i^2) - a_i b_i$ is positive. Completing the square completes the proof:

$$\frac{1}{2}(a_i^2 + b_i^2) - a_i b_i = \left(\frac{1}{\sqrt{2}} a_i - \frac{1}{\sqrt{2}} b_i \right)^2 \geq 0 \,. \tag{2.8}$$

□

This lemma brings us almost all the way to the approximation guarantee for Algorithm 3. The proof below completes it.

Theorem 26. RANDOMIZED-USM *(Algorithm 3) is a $\frac{1}{2}$-approximation for the unconstrained submodular maximization problem (Problem 5).*

Proof. Summing Lemma 25 over $i \in \{1, \ldots, N\}$, every other term cancels, leaving:

$$\mathbb{E}[f(\text{OPT}_0) - f(\text{OPT}_N)] \leq \frac{1}{2} \mathbb{E}[f(X_N) - f(X_0) + f(Y_N) - f(Y_0)] \,. \tag{2.9}$$

We can drop the X_0 and Y_0 terms here and maintain the inequality, as Problem 5 assumes $f(\emptyset)$ and $f(\Omega)$ are non-negative. Finally, recalling that $\text{OPT}_0 = \text{OPT}$, $\text{OPT}_N = X_N$, and $Y_N = X_N$, the inequality simplifies to:

$$\mathbb{E}[f(\text{OPT}) - f(X_N)] \leq \frac{1}{2} \mathbb{E}[2 f(X_N)] \implies \frac{1}{2} \mathbb{E}[f(\text{OPT})] \leq \mathbb{E}[f(X_N)] \,. \tag{2.10}$$

Thus, in expectation, $f(\text{OPT})$ is no more than twice $f(X_N)$. □

2.3 Extensions for Max-SAT and Max-Welfare

Algorithm 3 provides a tight approximation to Problem 5, the unconstrained submodular maximization problem. For more restricted classes of objective functions though, the approximation factor can be improved. Buchbinder et al. (2012) show that with very minor modifications, their algorithm has even better performance guarantees for submodular max-SAT and 2-player submodular max-welfare.

Problem 27. (Submodular max-SAT): *We are given a CNF formula composed of $\mathcal{C}$ clauses over N variables and a normalized, monotone submodular function $f : 2^{\mathcal{C}} \to \mathbb{R}_+$. The goal is to find an assignment to the variables, $\phi : \Omega \to \{0, 1\}$, maximizing $f(C(\phi))$, where $C(\phi) \subseteq \mathcal{C}$ are the satisfied clauses.*

Problem 28. (Submodular max-welfare): *We are given N items and k players, each with a normalized, monotone submodular function $f_j : 2^{\mathcal{N}} \to \mathbb{R}_+$. The goal is to divide Ω among the players, into partitions $\Omega_1, \ldots, \Omega_k$, while maximizing total welfare: $\sum_{j=1}^{k} f_j(\Omega_j)$.*

For submodular max-SAT and 2-player submodular max-welfare, Buchbinder et al. (2012) show a $\frac{3}{4}$-approximation. They do so for max-SAT by first defining an extended variable assignment ϕ' that can associate with each variable not only $\{0\}$ or $\{1\}$, but also both at once: $\{0, 1\}$. The set Y_0 then consists of all possible assignments to all variables. The $\frac{3}{4}$-approximation proof is nearly identical to that of Theorem 26, but invokes special knowledge about the value of Y_0 beyond just its non-negativity. Specifically, whereas the proof of Theorem 26 simply discards $f(Y_0)$, the proof for the special case of submodular max-SAT notes that $f(Y_0) \geq f(\text{OPT})$ and plugs this value into the final line of the proof.

For 2-player submodular max-welfare, the $\frac{3}{4}$-approximation guarantee is based on a reduction to the submodular max-SAT problem. Roughly, $2|\Omega|$ singelton clauses are created, one with each positive literal and one with each negative literal. Then, the utility function for player 1 is based on the positive clauses, P, and the utility function for player 2 is based on the negative clauses, $\mathcal{C} \setminus P$. Specifically: $f(C) = f_1(C \cap P) + f_2(C \cap (\mathcal{C} \setminus P))$.

According to hardness results proved in Vondrák (2009), a $\frac{3}{4}$-approximation is tight for both submodular max-SAT (Problem 27) and 2-player submodular max-welfare (Problem 28).

2.4 Discussion and Experiments

Buchbinder et al. (2012)'s algorithms yield many tight approximations, and this is certainly not their only attractive quality. They are also advantageous because of their simplicity, which makes them trivial to code, and their relatively low time complexity. Each algorithm makes $O(N)$ calls to the f-oracle, and the constant hidden by the big-O notation here is at most 4, even for a naïve implementation. Compared to other algorithms, such as those in Feldman et al. (2011b), this can be quite a bit faster for some submodular functions. (See the definition of the multilinear extension in Chapter 3; for functions whose multilinear extension must be approximated by sampling, algorithms that rely on this extension can become time-consuming.)

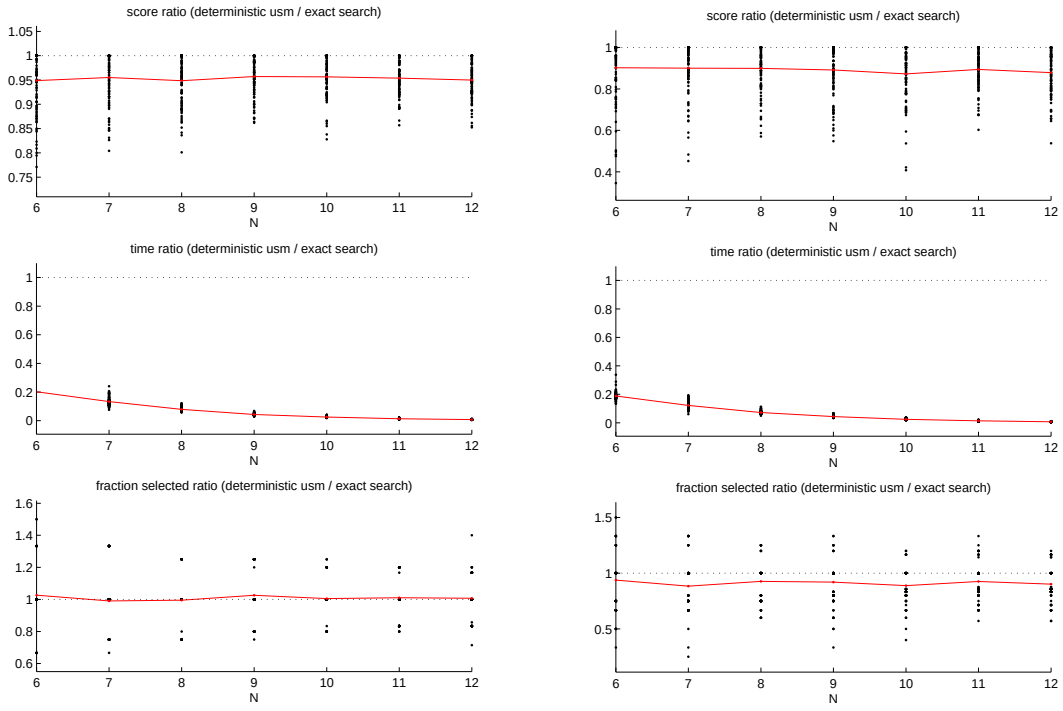
Unfortunately, Buchbinder et al. (2012) do not provide any empirical results. Therefore, to get a better feel for their algorithms' practical performance, we ran a few of our own small experiments. Although we tested with both the deterministic and randomized algorithms, we verified that in practice (at least in the settings we tested) there is no appreciable difference. Thus, only results for the deterministic variant are shown below. We investigated two non-monotone problems: graph cut (Problem 16) and log determinant (Problem 19). Note that log determinant does not always satisfy the non-negativity assumption, as $\log\det(L) \leq 0$ is possible. In the most extreme case, when L has two linearly dependent rows or columns, $\log\det(L) = -\infty$.

Graph cut setup: To generate inputs for the graph cut problem, we used the Watts-Strogatz random graph model (Watts and Strogatz, 1998). While relatively simple, this model nevertheless provides graphs that are typically much closer to real-world networks than those of the classical Erdős-Rényi graph model. Specifically, the Watts-Strogatz model yields relatively high clustering coefficients and a degree distribution

that follows a power law. The initial node degree we used with Watts-Strogatz was 5, and the rewiring probability was 0.1. After graph construction, each edge e was given a weight drawn uniformly from the interval $[0, 1]$: $w_e \sim \text{Uniform}(0, 1)$.

Log determinant setup: To generate inputs for the log determinant problem, we needed a source of positive semi-definite matrices. The Wishart distribution is a reasonable source in that it places substantial mass on matrices with diverse eigenvalues. This diversity implies that the subset selection problem is likely to be non-trivial. The specific Wishart we used had N degrees of freedom and a uniform, diagonal covariance matrix with scale 2: $\mathcal{W}_N(N, 2I)$. To keep the selected set size relatively distant from 0 and N, we further scaled all entries of the resulting Wishart matrix by $\frac{1}{N}$.

Figure 2.1 shows the results of small-scale experiments, with $N \leq 12$. In this realm, it is possible to run an exact search for the single best set. Notice that even at these small N there is a noticeable score loss compared to exact search. However, in terms of speed, exact search is a poor choice even at $N = 6$. Most interestingly, the graph at the lower right indicates that Algorithm 2 routinely under-selects for the log determinant objective. Additional experiments restricting to L with $\log\det(L) \geq 0$ show no such under-selection bias. This suggests that perhaps when designing algorithms for objectives that violate the non-negativity assumption, some further mechanism may be necessary to prevent under-selection.

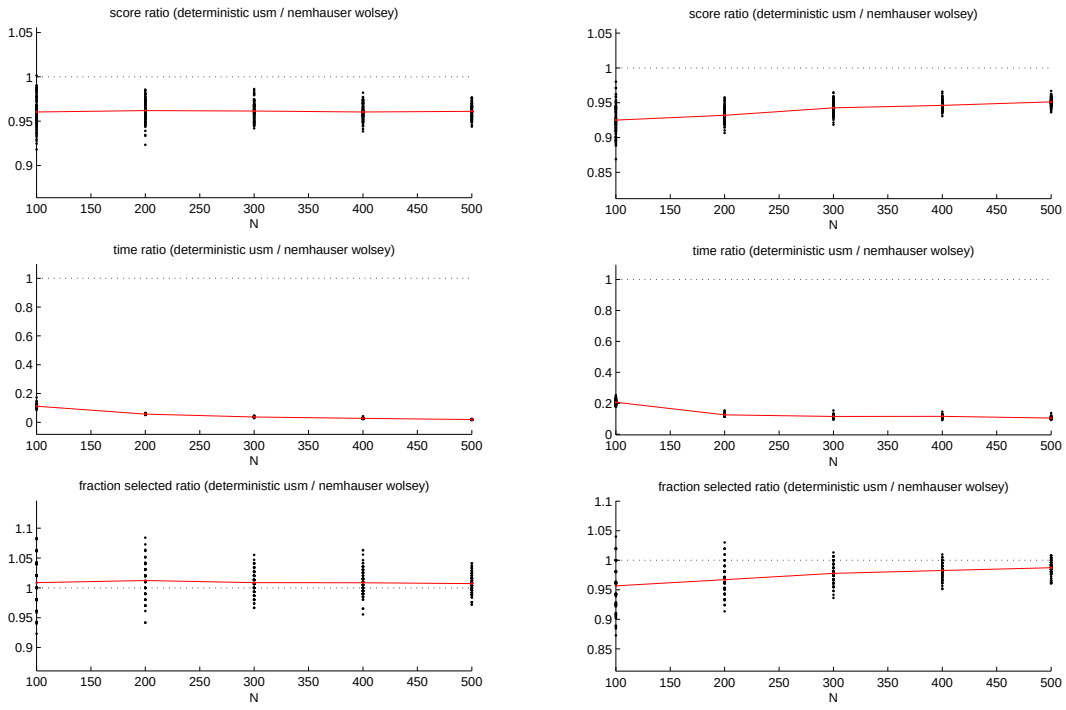


(a) (b)

Figure 2.1: Comparison to exact search for small N. These graphs show statistics for selected sets S from 100 random trials: score $f(S)$ (top), runtime (middle), and selection size $|S|$ (bottom). (a) Graph cut results. (b) Log determinant results. Dotted lines at 1 indicate equal performance. Red lines are the means. Black dots represent the individual random trials.

Figure 2.2 shows the results of larger-scale experiments, for N as large as 500. Since it isn't feasible to compare to exact search in this realm, we instead compare to the greedy algorithm of Nemhauser et al.

(1978), introduced earlier in this survey as Algorithm 1. Recall that Algorithm 1 is a $(1-1/e)$-approximation for normalized, non-negative, monotone submodular functions, but has no guarantees for non-monotone problems such as graph cut and log determinant. Nevertheless, in practice we see that it tends to out-score Algorithm 2. Algorithm 2 is at least faster, but for many practical problems, log determinant included, switching to a less naïve implementation of Algorithm 1 can reverse this advantage. ("Less naïve" here simply implies storing a few additional quantities across iterations to speed f-value queries.) Thus, the speed advantage of Algorithm 2 is somewhat questionable.



(a) (b)

Figure 2.2: Comparison to Algorithm 1 for N up to 500. These graphs show statistics on selected sets S from 100 random trials: score $f(S)$ (top), runtime (middle), and selection size $|S|$ (bottom). (a) Graph cut results. (b) Log determinant results. Dotted lines at 1 indicate equal performance. Red lines are the means. Black dots represent the individual random trials.

In conclusion, Buchbinder et al. (2012)'s algorithms represent a significant step forward theoretically, but may not often be the best choice in practice. Of course, in practice we also often find ourselves in a more constrained setting. The following chapters explore algorithms for these constrained settings.

Chapter 3

Cardinality-Constrained Setting

There are two basic types of cardinality constraints, the "capped" type, $|S| \leq k$, and the "fixed" type, $|S| = k$. Both are in fact examples of constraints derived from the simple uniform matroid defined in Section 1.2.3. The "capped" type results from restricting S to the independent sets of the matroid, while the "fixed" type comes from restricting S further, to the bases of the matroid.

Problem 29. (Cardinality-capped submodular maximization) *For a submodular function* $f : 2^{\Omega} \to \mathbb{R}_+$, *find:* $\max_{S \subseteq \Omega : |S| \leq k} f(S)$.

Problem 30. (Cardinality-fixed submodular maximization) *For a submodular function* $f : 2^{\Omega} \to \mathbb{R}_+$, *find:* $\max_{S \subseteq \Omega : |S| = k} f(S)$.

The work of Buchbinder et al. (2014) provides the current best guarantees for maximizing non-monotone submodular functions subject to either of these cardinality constraints. For $|S| \leq k$, their work provides approximation factors in the $[\frac{1}{e} + 0.004, \frac{1}{2} - o(1)]$ range. For $|S| = k$, the range achieved is almost as good: $[0.356, \frac{1}{2} - o(1)]$. In both settings, the algorithms are $(\frac{1}{2} - o(1))$-approximations at $k = \frac{N}{2}$. For this particular k, no better approximation factor is possible — Corollaries 5.5 and 5.9 of Vondrák (2009) establish that a $(\frac{1}{2} + \varepsilon)$-approximation would require an exponential number of queries to an f-oracle.

To give a sense of how the approximation factors change with k, Figure 3.1 plots the exact approximation guarantees for the full range of possible k, ignoring the $o(1)$ term. The plots show that the approximation factor for $|S| = k$ is symmetric about $k = \frac{N}{2}$, while for $|S| \leq k$ it increases monotonically with k. In both settings, the plots exhibit a piecewise nature. This is due to the fact that instead of relying on a single algorithm, Buchbinder et al. (2014)'s guarantees come from running two algorithms and returning the max-scoring result. The remainder of this chapter focuses on these algorithms and their associated proofs.

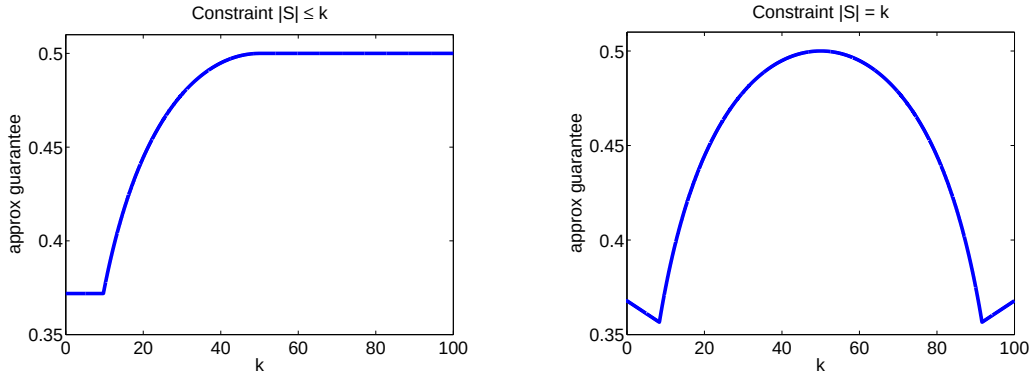


Figure 3.1: Best approximation guarantees of Buchbinder et al. (2014) for $N = 100$.

3.1 Selected Algorithms

The two core algorithms Buchbinder et al. (2014) rely on are shown here as Algorithm 4, RANDOM-GREEDY, and Algorithm 5, CONTINUOUS-DOUBLE-GREEDY. The first, RANDOM-GREEDY, is very similar to NEMHAUSER-WOLSEY (Algorithm 1). The main difference is a little randomization; instead of selecting the highest-scoring element, RANDOM-GREEDY selects randomly among the top k highest-scoring elements. Notice also that the M_i selection line of RANDOM-GREEDY doesn't make sense for $k > \frac{N}{2}$. That is, the set U_i would contain fewer than the necessary k elements for iterations $i > \frac{N}{2}$. This issue can be easily handled for both types of cardinality constraint though, using Reductions 31 and 32.

Reduction 31. *For the $|S| \leq k$ constraint and $k > \frac{N}{2}$, add $2k$ "dummy" elements, D, to the ground set: $\Omega' = \Omega \cup D$. The value of any set $S \subseteq \Omega'$ does not depend on D: $f(S) = f(S \setminus D)$. If any dummy elements are selected over the course of the algorithm, they can simply be discarded from the returned set without changing its f-value.*

Reduction 32. *For the $|S| = k$ constraint and $k > \frac{N}{2}$, set $\bar{k} = N - k$ and let $\bar{f}(S) = f(\Omega \setminus S)$. Note that, by Theorem 4, $\bar{f}$ is submodular. Then maximizing $\bar{f}$ constrained to $|S| = \bar{k}$ is equivalent to maximizing f subject to $|S| = k$. Since $\bar{k} = N - k < N - \frac{N}{2} \leq \frac{N}{2}$, the $k > \frac{N}{2}$ issue is resolved.*

The "dummy" items of Reduction 31 also seem to be necessary whenever there are not enough real items with non-negative marginal gain to populate M_i. That is, whenever there are fewer than k items remaining whose addition to S_{i-1} would increase f-value. The non-negativity of these marginal gains appears to be relied upon in some of the proofs (see Equation (3.7) in Lemma 38).

The second main algorithm that Figure 3.1's guarantees rely on is CONTINUOUS-DOUBLE-GREEDY (Algorithm 5). This is also similar to algorithms previously discussed: DETERMINISTIC-USM and RANDOMIZED-USM from Section 2.1. CONTINUOUS-DOUBLE-GREEDY represents an extension of these more basic algorithms to the continuous setting. In this setting, it becomes possible to incorporate cardinality constraints while maintaining an approximation guarantee.

More precisely, CONTINUOUS-DOUBLE-GREEDY relies on an intuitive continuous extension of f, called the mutilinear extension. This extension was first introduced by Calinescu et al. (2007), and we will encounter it again in Chapter 4. For a vector $\boldsymbol{x} \in [0,1]^\Omega$, consider a random subset $R(\boldsymbol{x})$ that contains element i of Ω with probability x_i. The multilinear extension is the expected value of f on $R(\boldsymbol{x})$: $F(\boldsymbol{x}) = \mathbb{E}[f(R(\boldsymbol{x}))]$. More formally:

Definition 33. (Multilinear extension) *The multilinear extension of $f : 2^\Omega \to \mathbb{R}$ is $F : [0,1]^\Omega \to \mathbb{R}$ with value:*

$$F(\boldsymbol{x}) = \sum_{S \subseteq \Omega} f(S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i) \,. \tag{3.1}$$

Notice that explicitly computing F according to this formula would require summing over 2^N sets $S \subseteq \Omega$. For some f though, such as graph cuts, F can be computed efficiently. For all other f, by randomly sampling sets S according to the probabilities in $\boldsymbol{x}$, $F(\boldsymbol{x})$ can be estimated arbitrarily well. Calinescu et al. (2007) state that for any f, a polynomial number of samples yields a $(1 - 1/\text{poly}(N))$-approximation to $F(\boldsymbol{x})$.

Before considering associated proofs, we pause to make a few observations about how CONTINUOUS-DOUBLE-GREEDY can be implemented in practice. The line "at any time $t \in [0,1]$" really means that t serves as a form of iteration counter. This line can be replaced by initializing $t = 0$, running "while $t < 1$", and updating $t \leftarrow t + \delta$ for some small step size δ each iteration. The $\boldsymbol{x}$ and $\boldsymbol{y}$ values can then be similarly updated, setting $x_i^{t+\delta} \leftarrow x_i^t + \delta \frac{dx_i}{dt}(\ell^*)$. Additionally, the step of finding ℓ' simplifies to searching over a relatively small set of possibilities, since Buchbinder et al. (2014) show $\boldsymbol{x}$'s derivative is restricted to three settings:

$$\frac{dx_i}{dt} = \begin{cases} 1 & \ell < -b_i \\ \frac{a_i - \ell}{a_i + b_i} & -b_i \leq \ell \leq a_i \\ 0 & \ell > a_i \,. \end{cases} \tag{3.2}$$

Finally, there is the issue of converting the output $\boldsymbol{x}^1$ from a $[0,1]$ vector to a subset of Ω. This can be done without loss in expected f-value, by using pipage rounding techniques (see Section 2.4 of Calinescu et al. (2011)).

Algorithm 4 RANDOM-GREEDY

Input: submodular function f, ground set Ω, limit k
$S_0 \leftarrow \emptyset, U_0 \leftarrow \Omega$
for $i = 1$ **to** k **do**
  for $j \in U_{i-1}$ **do**
    $a_j \leftarrow f(S_{i-1} \cup \{j\}) - f(S_{i-1})$
  end for
  $M_i \leftarrow \arg\max_{J \subseteq U_{i-1}:|J|=k} \sum_{j \in J} a_j$
  $u_i \sim \text{Uniform}(M_i)$
  $S_i \leftarrow S_{i-1} \cup \{u_i\}$
  $U_i \leftarrow U_{i-1} \setminus \{u_i\}$
end for
return S_k

Algorithm 5 CONTINUOUS-DOUBLE-GREEDY

Input: multilinear extension F, ground set Ω, limit k
$\boldsymbol{x}^0 \leftarrow \emptyset, \boldsymbol{y}^0 \leftarrow \Omega$
at any time $t \in [0,1]$ **do**
  define for every $i \in \Omega$ **:**
    $a_i \leftarrow \frac{\partial F(\boldsymbol{x}^t)}{\partial x_i}$, $b_i \leftarrow -\frac{\partial F(\boldsymbol{y}^t)}{\partial y_i}$
    $a_i'(\ell) \leftarrow \max\{a_i - \ell, 0\}$, $b_i'(\ell) \leftarrow \max\{b_i + \ell, 0\}$
    $\frac{dx_i}{dt}(\ell) = \frac{a_i'}{a_i'+b_i'}$, $\frac{dy_i}{dt}(\ell) = \frac{b_i'}{a_i'+b_i'}$
  Set ℓ' such that $\sum_{i \in \Omega} \frac{dx_i}{dt}(\ell') = k$
  If enforcing $|S| = k$: $\ell^* = \ell'$
  If enforcing $|S| \leq k$: $\ell^* = \max\{\ell', 0\}$
  for $i \in \Omega$ **do**
    $\frac{dx_i}{dt} \leftarrow \frac{dx_i}{dt}(\ell^*)$, $\frac{dy_i}{dt} \leftarrow \frac{dy_i}{dt}(\ell^*)$
  end for
return $\boldsymbol{x}^1$

3.2 Selected Proofs

The main approximation guarantees of Buchbinder et al. (2014) are based on combinations of algorithms. Still, each algorithm by itself has some useful guarantees as well. The next two sections address each algorithm in isolation, and the third section discusses their combination.

3.2.1 Random Greedy Proofs

Recall that RANDOM-GREEDY is a randomized version of NEMHAUSER-WOLSEY. It turns out that this randomization doesn't hurt the approximation guarantee associated with NEMHAUSER-WOLSEY. This means that RANDOM-GREEDY is in fact a $(1-1/e)$-approximation for *monotone* submodular functions subject to the capping constraint $|S| \leq k$. (Note that for monotone functions $|S| \leq k$ and $|S| = k$ are essentially identical constraints, since it never hurts the f-value to select more items.)

Unlike NEMHAUSER-WOLSEY though, RANDOM-GREEDY has guarantees for *non-monotone* functions as well. Neither of these guarantees is as good, for all k, as the guarantees shown in Figure 3.1, but RANDOM-GREEDY does have a time complexity advantage over the procedure required to get the Figure 3.1 guarantees.

Theorem 34. *Algorithm 4 is a $\frac{1}{e}$-approximation for the cardinality-capped submodular maximization problem (Problem 29).*

Theorem 35. *Algorithm 4 is a $((1 - \frac{k}{eN})/e - \varepsilon)$-approximation for the cardinality-fixed submodular maximization problem (Problem 30), assuming $k \leq \frac{N}{2}$. The same applies for $k > \frac{N}{2}$, replacing k with $n - k$ in the approximation factor. The ε is an arbitrarily small positive constant.*

Figure 3.2 plots these approximation guarantees for the full range of possible k, ignoring ε. Notice that the $|S| = k$ plot here exactly matches the $|S| = k$ plot of Figure 3.1 in the low-k and high-k regions. Similarly, the $|S| \leq k$ plot here nearly matches the $|S| \leq k$ plot of Figure 3.1 in the low-k region. The match is not quite perfect in this second case though; recall that the approximation factor foreshadowed in the

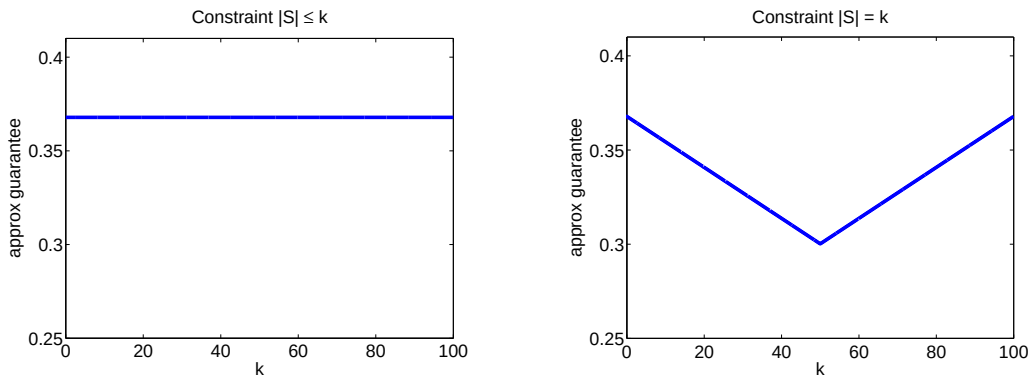


Figure 3.2: Approximation guarantees of Theorem 34 and Theorem 35 for $N = 100$.

beginning of this chapter had minimum value $\frac{1}{e} + 0.004$. Obtaining the additional 0.004 is discussed later, in Section 3.2.3.

The proofs for Theorem 34 and Theorem 35 rely on many similar lemmas and techniques. The cardinality-capped one is simpler though, so we describe just that one here. It relies on three supporting lemmas. The proof of the first lemma is a fairly straightforward application of the definition of submodularity, so we simply state the first lemma without proof.

Lemma 36. *Let $A(p)$ represent a random subset of A where each element is included with probability* at most *p. Then, for non-negative, submodular f, we have: $\mathbb{E}[f(A(p))] \geq (1-p)f(\emptyset)$.*

Lemma 37. *For $i \in \{0, \ldots, k\}$, it is the case that: $(1-1/k)^i f(\text{OPT}) \leq \mathbb{E}[f(\text{OPT} \cup S_i)]$.*

Proof. Even if an element ends up in M_i, its chance of being left out of S_i is $1-1/k$. The chance of being left out for all $j \leq i$ is thus at least $(1-1/k)^i$. So, the chance that an element is in S_i is at most $1-(1-1/k)^i$. Define an (also submodular) variation on f: $g(S) = f(S \cup \text{OPT})$. Then, applying Lemma 36 to g yields the desired result:

$$\mathbb{E}[f(\text{OPT} \cup S_i)] = \mathbb{E}[g(S_i \setminus \text{OPT})] \geq (1-1/k)^i \, g(\emptyset) = (1-1/k)^i \, f(\text{OPT}) \, . \tag{3.3}$$

□

Lemma 38. *For $i \in \{0, \ldots, k\}$, it is the case that:*

$$\frac{1}{k}\left(1-\frac{1}{k}\right)^{i-1} f(\text{OPT}) - \frac{1}{k}\mathbb{E}[f(S_{i-1})] \leq \mathbb{E}[f(S_{i-1} \cup \{u_i\}) - f(S_{i-1})] \, . \tag{3.4}$$

Proof. First, fix an event A_{i-1} that covers all random decisions up to iteration i. Conditioned on this, S_{i-1} is also fixed. Recall that u_i represents the element selected at iteration i:

$$\mathbb{E}[f(S_{i-1} \cup \{u_i\}) - f(S_{i-1})] = \frac{1}{k} \sum_{j \in M_i} (f(S_{i-1} \cup \{j\}) - f(S_{i-1})) \, . \tag{3.5}$$

Also recall that the elements in M_i are those with the largest marginal gains. Thus, for any other size-k subset $M_i' \subseteq U_{i-1}$, the gains are smaller:

$$\mathbb{E}[f(S_{i-1} \cup \{u_i\}) - f(S_{i-1})] \geq \frac{1}{k} \sum_{j \in M_i'} (f(S_{i-1} \cup \{j\}) - f(S_{i-1})) \, . \tag{3.6}$$

The set $\text{OPT} \setminus S_{i-1}$ is a subset of U_{i-1} and can be made size-k by the addition of dummy elements (see Reduction 31). Substituting this set for M_i' and applying the definition of submodularity:

$$\mathbb{E}[f(S_{i-1} \cup \{u_i\}) - f(S_{i-1})] \geq \frac{1}{k} \sum_{j \in \text{OPT} \setminus S_{i-1}} (f(S_{i-1} \cup \{j\}) - f(S_{i-1})) \geq \frac{1}{k}(f(\text{OPT} \cup S_{i-1}) - f(S_{i-1})) \,. \quad (3.7)$$

Now taking the expectation over all random decisions before iteration i and applying Lemma 37:

$$\mathbb{E}[f(S_{i-1} \cup \{u_i\}) - f(S_{i-1})] \geq \frac{1}{k}\mathbb{E}[f(\text{OPT} \cup S_{i-1}) - f(S_{i-1})] \geq \frac{1}{k}\left(1 - \frac{1}{k}\right)^{i-1} f(\text{OPT}) - \frac{1}{k}\mathbb{E}[f(S_{i-1})] \,. \quad (3.8)$$

□

Given this final supporting lemma, the proof of Theorem 34 is now possible.

Proof. **(Of Theorem 34)** We will show by induction on i that $\mathbb{E}[f(S_i)] \geq \frac{i}{k}(1 - \frac{1}{k})^{i-1} f(\text{OPT})$.

- Base case: For $i = 0$, $\mathbb{E}[f(S_0)] = f(\emptyset)$. Since we assume in the statement of Problem 29 that f is non-negative, $f(\emptyset) \geq 0$.
- Inductive case: The inductive hypothesis holds for every $i' \leq i$. For i itself:

$$\begin{aligned}
\mathbb{E}[f(S_i)] &= \mathbb{E}[f(S_{i-1} \cup \{u_i\})] && (3.9)\\
&= \mathbb{E}[f(S_{i-1})] + \mathbb{E}[f(S_{i-1} \cup \{u_i\}) - f(S_{i-1})] && (3.10)\\
&\quad \text{Adding and subtracting } f(S_{i-1}).\\
&\geq \mathbb{E}[f(S_{i-1})] + (1/k)\,(1 - 1/k)^{i-1}\, f(\text{OPT}) - (1/k)\mathbb{E}[f(S_{i-1})] && (3.11)\\
&\quad \text{Applying Lemma 38.}\\
&= (1 - 1/k)\mathbb{E}[f(S_{i-1})] + (1/k)\,(1 - 1/k)^{i-1}\, f(\text{OPT}) && (3.12)\\
&\quad \text{Re-arranging terms.}\\
&\geq (1 - 1/k)\frac{i-1}{k}(1 - 1/k)^{i-2} f(\text{OPT}) + (1/k)\,(1 - 1/k)^{i-1}\, f(\text{OPT}) && (3.13)\\
&\quad \text{Applying the inductive hypothesis to } S_{i-1}.\\
&= (i/k)(1 - 1/k)^{i-1} f(\text{OPT}) \,. && (3.14)\\
&\quad \text{Summing the coefficients.}
\end{aligned}$$

Evaluating at $i = k$, we have the desired result: $\mathbb{E}[f(S_k)] \geq \frac{k}{k}\left(1 - \frac{1}{k}\right)^{k-1} f(\text{OPT}) \geq \frac{1}{e} f(\text{OPT})$. □

3.2.2 Continuous Double Greedy Proofs

The CONTINUOUS-DOUBLE-GREEDY algorithm also provides approximation guarantees for cardinality-constrained submodular maximization.

Theorem 39. *Algorithm 5 achieves an approximation factor of:*

$$\left(1 + \frac{N}{2\sqrt{(N-k)k}}\right)^{-1} - o(1) \quad (3.15)$$

for cardinality-capped and cardinality-fixed submodular maximization (Problems 29 and 30), for $k \leq \frac{N}{2}$. For $k \geq \frac{N}{2}$, the approximation factor remains unchanged for Problem 30, but saturates at $\frac{1}{2}$ for Problem 29.

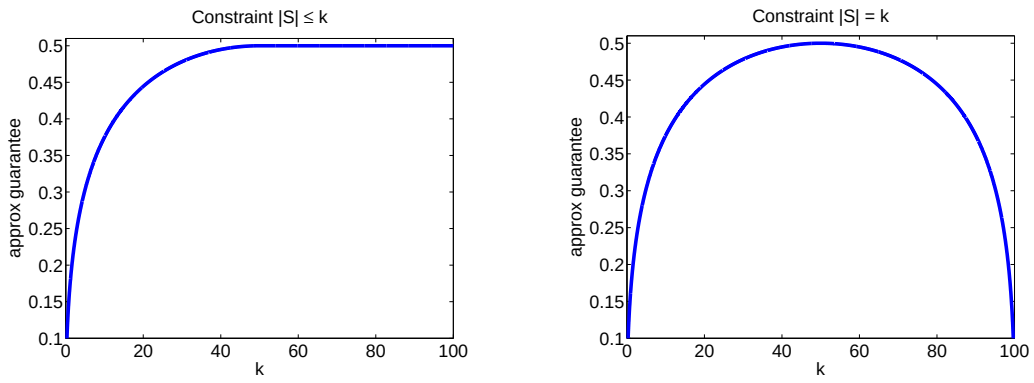


Figure 3.3: Approximation guarantees of Theorem 39 for $N = 100$.

Figure 3.3 plots these approximation guarantees for the full range of possible k, ignoring the $o(1)$ term. Notice that the $|S| \leq k$ plot here matches the $|S| \leq k$ plot of Figure 3.1 everywhere except in the low-k region. Similarly, the $|S| = k$ plot here matches the $|S| = k$ plot of Figure 3.1 everywhere except in the low-k and high-k regions. In these excepted regions the RANDOM-GREEDY algorithm has a better guarantee.

The proof of Theorem 39 is somewhat involved so we omit it here, but note that it essentially contains many of the same steps as the proof of Theorem 26 from the previous chapter. For example, the $a_i + b_i \geq 0$ property from Lemma 22 is proven again. Then, a quantity analogous to the $\text{OPT}_i = (\text{OPT} \cup X_i) \cap Y_i$ set from Lemma 25 is defined and its change across iterations is again upper-bounded. More concretely, this quantity is the vector $(\text{OPT} \vee \boldsymbol{x}^t) \wedge \boldsymbol{y}^t$, where $\vee$ is the coordinate-wise maximum and $\wedge$ is the coordinate-wise minimum. Instead of summing over all iterations, as was done for Theorem 26, the final step in the continuous proof is to integrate from $t = 0$ to $t = 1$.

3.2.3 Combined Algorithm Proofs

For the $|S| = k$ case, we have already covered all the machinery necessary to state the final approximation guarantees that were foreshadowed at the beginning of this chapter.

Theorem 40. *Running Algorithm 4 (*RANDOM-GREEDY*) and Algorithm 5 (*CONTINUOUS-DOUBLE-GREEDY*) and taking the better of the two outputs yields an approximation factor of:*

$$\max\left\{\frac{(1-\frac{k}{eN})}{e} - \varepsilon, \left(1 + \frac{N}{2\sqrt{(N-k)k}}\right)^{-1} - o(1)\right\} \tag{3.16}$$

for the cardinality-fixed submodular maximization problem (Problem 30).

Proof. Taking the max of the approximation factors from Theorem 35 and Theorem 39, the result is immediate. □

For the $|S| \leq k$ case, we nearly have all the machinery for the foreshadowed guarantee, but require an additional 0.004, to make $\frac{1}{e} + 0.004$. To break the $\frac{1}{e}$ barrier, Buchbinder et al. (2014) create a variation on RANDOM-GREEDY called WIDE-RANDOM-GREEDY. This algorithm that is identical to RANDOM-GREEDY except that it sometimes chooses the next element to add, u_i, from a set larger than size k. More precisely, the set M_i is of size $\Sigma(i)$:

$$\Sigma(i) = \begin{cases} 2(k-i+1) & i \leq \lceil 0.21k \rceil \\ k & i > \lceil 0.21k \rceil \end{cases}. \tag{3.17}$$

This WIDE-RANDOM-GREEDY algorithm gives the extra 0.004 to the approximation guarantee, but only when a certain assumption on the worst element in M_i is satisfied. To guarantee the extra 0.004 in all

cases, it is necessary to also run the AUGMENTATION-PROCEDURE, shown as Algorithm 6 here, for every $i \in \{1, \dots, \lceil 0.21k \rceil\}$. The details of the proof are beyond the scope of this survey, but we state the main result as Theorem 41.

Theorem 41. *Running* WIDE-RANDOM-GREEDY *and* AUGMENTATION-PROCEDURE *(Algorithm 6) and taking the best of their outputs yields an approximation factor of:*

$$\max \left\{ \frac{1}{e} + 0.004, \left(1 + \frac{N}{2\sqrt{(N-k)k}} \right)^{-1} - o(1) \right\} \tag{3.18}$$

for the cardinality-capped submodular maximization problem (Problem 29) and $k \leq \frac{N}{2}$. *For* $k > \frac{N}{2}$, *the approximation factor is* $\frac{1}{2}$.

Algorithm 6 AUGMENTATION-PROCEDURE

Input: submodular f, ground set Ω, limit k, iteration count i, set S_{i-1} from WIDE-RANDOM-GREEDY
$M_i \leftarrow \underset{J \subseteq \Omega \setminus S_{i-1} : |J| = 2(k-i+1)}{\arg\max} \sum_{j \in J} f(S_{i-1} \cup \{j\})$
Define the submodular function $g_i(B) = f(B \cup S_{i-1})$ and multilinear extension G_i
$B_i \leftarrow$ CONTINUOUS-DOUBLE-GREEDY$(G_i, M_i, k - i + 1)$
return $S_{i-1} \cup B_i$

3.3 Discussion

The Buchbinder et al. (2014) algorithms discussed in this chapter provide the current best guarantees for maximizing non-monotone submodular functions subject to cardinality constraints. Buchbinder et al. (2014) also show that a variation on RANDOM-GREEDY (Algorithm 4) can handle any single matroid constraint while providing an approximation factor of $\frac{1}{4}$. In practical applications though, the RANDOM-GREEDY algorithm is unlikely to perform better than the NEMHAUSER-WOLSEY algorithm (Algorithm 1); the two only differ in that RANDOM-GREEDY is more random. End-users of submodular maximization algorithms will probably regard the proofs surrounding RANDOM-GREEDY as an indirect justification for using its close neighbor, NEMHAUSER-WOLSEY, for non-monotone objectives.

The CONTINUOUS-DOUBLE-GREEDY algorithm (Algorithm 5) may have greater practical importance. Recall the empirical results for its discrete relatives from the previous chapter, DETERMINISTIC-USM (Algorithm 2) and RANDOMIZED-USM (Algorithm 3). The f-values achieved were not impressive compared to those of NEMHAUSER-WOLSEY, and it is debatable how much the change from discrete to continuous would improve these scores. However, in terms of runtime, the continuous algorithm could sometimes have a substantial edge. The ideal setting for its use would be with functions f whose multilinear extension F can be computed efficiently and with test cases where running all the way to $t = 1$ isn't necessary for convergence of $\boldsymbol{x}$. Note that there are many F that can be computed efficiently, including for graph cut application from the previous chapter: for a graph with adjacency matrix A, summing the entries of the matrix $B = \text{diag}(\boldsymbol{x})\ A\ \text{diag}(1 - \boldsymbol{x})$ gives the value of F: $F(\boldsymbol{x}) = \sum_{i,j} B_{ij}$.

Besides the practicality of these algorithms, it is important to consider what they suggest in terms of future theoretical work. The breaking of the "natural" $\frac{1}{e}$ barrier in Theorem 41 suggests that we don't yet know the best algorithm for all cardinalities. As mentioned earlier, the approximation is tight for $k = \frac{N}{2}$, but for smaller k it is possible that tighter algorithms or proofs exist.

In the following chapter we explore algorithms for other constraint settings.

Chapter 4

Polytope-Constrained Setting

In the previous chapter we briefly discussed that simple cardinality constraints, $|S| \leq k$ and $|S| = k$, can be thought of as types of matroid constraints. The $|S| \leq k$ constraint results from restricting S to the independent sets of the uniform matroid, while the $|S| = k$ constraint comes from restricting S further, to the bases of the matroid. These restrictions can also be characterized in terms of polytopes. To illustrate this, we first introduce a few more technical definitions.

Definition 42. (Incidence vector) *Let $\boldsymbol{e}_i \in \mathbb{R}^N$ denote the standard unit vector with a 1 in entry i and zeros elsewhere. For a set S, its associated incidence vector is $e_S = \sum_{i \in S} \boldsymbol{e}_i$.*

Definition 43. (Basis matroid polytope) *For a matroid with bases $\mathcal{B}$, the polytope $P_{\mathcal{B}} = conv(e_B \mid B \in \mathcal{B})$ is called the basis matroid polytope. This is the convex hull of the incidence vectors of the bases.*

Definition 44. (Independence matroid polytope) *For a matroid with independent sets $\mathcal{I}$, the polytope $P_{\mathcal{I}} = conv(e_I \mid I \in \mathcal{I})$ is called the independence matroid polytope. (The basis matroid polytope is one face of the independence matroid polytope.)*

The cardinality constraint $|S| = k$ can be thought of as the restriction of S to the vertices of the uniform matroid's basis matroid polytope. The same applies for $|S| \leq k$, but for the independence matroid polytope. Going beyond cardinality, there are many other useful constraints that can be described in terms of polytopes. To begin with, there are all the other matroids besides the uniform one (see Section 1.2.3 for examples). Even outside the realm of matroids though, there are notable examples. For instance, knapsack constraints.

Definition 45. (Knapsack constraint) *A constraint of the form $\sum_{i=1}^{N} a_i x_i \leq b$ for non-negative values a_i, x_i, b is called a knapsack constraint.*

Definition 46. (Knapsack polytope) *For a given knapsack constraint, the convex hull of all the feasible solutions is called the knapsack polytope: conv $\left(\boldsymbol{x} \mid \sum_{i=1}^{N} a_i x_i \leq b\right)$.*

On top of all these examples, there is the fact that the intersection of two polytopes is itself a polytope. Significantly, this means that the intersection of any number of these constraints can be described as a polytope. Having established that polytope constraints are ubiquitous, it is clear that a submodular maximization algorithm general enough to handle a polytope constraint would be an extremely useful tool. For efficient algorithms, a simple restriction on the class of polytopes is first necessary: solvability.

Definition 47. (Solvable polytope) *A polytope $P \subseteq [0,1]^{\Omega}$ is solvable if for any linear objective function $g(\boldsymbol{x}) = \mathbf{a}^{\top}\boldsymbol{x}$, it is possible to efficiently find the best $\boldsymbol{x}$ in P: $\boldsymbol{x} = \arg\max_{\boldsymbol{y} \in P} g(\boldsymbol{y})$.*

For any *solvable* polytope P, Vondrák (2008)'s "continuous greedy" algorithm is a $(1-1/e)$-approximation for the problem $\max_{\boldsymbol{x} \in P} F(\boldsymbol{x})$, when f is monotone. (Recall that F is the multilinear extension of f; see Definition 33.) Unfortunately, these guarantees do not extend to non-monotone f. In fact, an additional

restriction on the class of polytopes is necessary for the non-monotone case — according to Vondrák (2009), no constant factor approximation is possible unless the polytope is down-monotone.

Definition 48. (Down-monotone polytope) *A polytope $P \subseteq [0,1]^\Omega$ is down-monotone if for all $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^\Omega$, $\boldsymbol{y} \leq \boldsymbol{x}$ and $\boldsymbol{x} \in P$ implies $\boldsymbol{y} \in P$. (Inequalities apply element-wise: $\boldsymbol{y} \leq \boldsymbol{x}$ implies $y_i \leq x_i$ for all i.)*

Fortunately, the class of solvable, down-monotone polytopes still includes all of the example polytopes given above, including matroid polytopes and knapsack polytopes. The remainder of this chapter focuses on the work of Chekuri et al. (2011), which provides approximation guarantees for algorithms that operate on non-monotone submodular f, constrained to *solvable, down-monotone* polytopes.

4.1 Multilinear Extension

Chekuri et al. (2011)'s algorithms rely heavily on the multilinear extension, $F(\boldsymbol{x})$, that was briefly introduced in the last chapter (see Definition 33). In particular, they focus on solving the following problem.

Problem 49. (Down-monotone polytope-constrained multilinear maximization) *For the multilinear extension $F : [0,1]^\Omega \to \mathbb{R}_+$ of a submodular function $f : 2^\Omega \to \mathbb{R}_+$, and a down-monotone polytope $P \subseteq [0,1]^\Omega$, find: $\max_{\boldsymbol{x} \in P} F(\boldsymbol{x})$.*

Chekuri et al. (2011)'s approximation guarantees for this problem depend on several basic properties of F. First, that F is "submodular" in a continuous sense.

Definition 50. (Continuous submodularity) *A function $G : [0,1]^\Omega \to \mathbb{R}$ is submodular if, for all $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^\Omega$:*

$$G(\boldsymbol{x} \vee \boldsymbol{y}) + G(\boldsymbol{x} \wedge \boldsymbol{y}) \leq G(\boldsymbol{x}) + G(\boldsymbol{y}) \tag{4.1}$$

where $(\boldsymbol{x} \vee \boldsymbol{y})_i = \max\{x_i, y_i\}$ and $(\boldsymbol{x} \wedge \boldsymbol{y})_i = \min\{x_i, y_i\}$.

It is also important that, while not concave in general, F is concave in any non-negative or non-positive direction.

Lemma 51. *The multilinear extension F is concave along any line with a non-negative direction vector. That is, any vector v with $v_i \geq 0 \ \ \forall i$.*

Proof. First, we show that the second derivatives of F are always negative.

$$\frac{\partial F(\boldsymbol{y})}{\partial y_j} = \mathbb{E}[f(\hat{\boldsymbol{y}}) \mid \hat{y}_j = 1] - \mathbb{E}[f(\hat{\boldsymbol{y}}) \mid \hat{y}_j = 0] \tag{4.2}$$

$$\frac{\partial^2 F(\boldsymbol{y})}{\partial y_i \partial y_j} = \mathbb{E}[f(\hat{\boldsymbol{y}}) \mid \hat{y}_i = 1, \hat{y}_j = 1] + \mathbb{E}[f(\hat{\boldsymbol{y}}) \mid \hat{y}_i = 0, \hat{y}_j = 0] \tag{4.3}$$

$$- \mathbb{E}[f(\hat{\boldsymbol{y}}) \mid \hat{y}_i = 1, \hat{y}_j = 0] - \mathbb{E}[f(\hat{\boldsymbol{y}}) \mid \hat{y}_i = 0, \hat{y}_j = 1] \leq 0 \tag{4.4}$$

where the last inequality follows by submodularity of F (Definition 50); for a vector $\boldsymbol{z} = [0,1]$ and a vector $\boldsymbol{w} = [1,0]$, we have the set $\boldsymbol{z} \vee \boldsymbol{w} = [1,1]$ and the set $\boldsymbol{z} \wedge \boldsymbol{w} = [0,0]$.

Now, let $\mathbf{d} \in [0,1]^\Omega$ be a positive direction vector, $\mathbf{d} \geq 0$, and let $t > 0$ be a scalar variable. We show that $\frac{\partial^2 F(\boldsymbol{x}+t\mathbf{d})}{\partial t^2} \leq 0$. In what follows, $\boldsymbol{y}$ serves as shorthand for the quantity $\boldsymbol{x} + t\mathbf{d}$.

$$\frac{\partial F(\boldsymbol{x} + t\mathbf{d})}{\partial t} = \sum_{i=1}^{N} d_i \frac{\partial F(\boldsymbol{y})}{\partial y_i} \tag{4.5}$$

$$\frac{\partial^2 F(\boldsymbol{x} + t\mathbf{d})}{\partial t^2} = \sum_{i=1}^{N} d_i \sum_{j=1}^{N} d_j \frac{\partial^2 F(\boldsymbol{y})}{\partial y_j \partial y_i} \leq 0 \tag{4.6}$$

where the last inequality follows by non-negativity of $\mathbf{d}$ and negativity of F's second derivatives. □

Corollary 52. *The multilinear extension F is concave along any line with a non-positive direction vector. That is, any vector v with $v_i \leq 0 \ \forall i$.*

Proof. This result follows immediately from the last equation in Lemma 51's proof. If both d_i and d_j there are negative, the signs cancel. □

Having established these properties of F, we illustrate them on one simple graph cut example before moving onto describe Chekuri et al. (2011)'s algorithms. Figure 4.1 shows a single-edge, two-node graph. The multilinear extension for this graph's cut function is plotted on the right. Notice that in general it is neither concave nor convex. Yet, along the line from $(0, 0)$ to $(1, 1)$ it is clearly concave. Similarly, it is concave in every other non-negative (or non-positive) direction.

$$
\begin{aligned}
F(\mathbf{x}) = {} & f(\emptyset)(1-x_1)(1-x_2)+ \\
& f(\{1\})x_1(1-x_2)+ \\
& f(\{2\})(1-x_1)x_2+ \\
& f(\{1,2\})x_1x_2 \\
= {} & x_1(1-x_2)+(1-x_1)x_2
\end{aligned}
$$

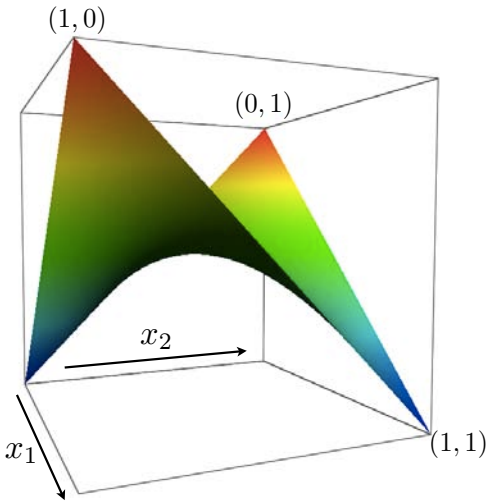


Figure 4.1: Multilinear extension for the cut function of 2-node graph.

4.2 Selected Algorithms

The core algorithm Chekuri et al. (2011) rely on is shown as Algorithm 7 here. We will refer to it as LOCAL-OPT. Essentially, this is a conditional gradient ascent algorithm (Frank and Wolfe, 1956). Observe that, by the down-monotone assumption, the starting point $\boldsymbol{x} = \mathbf{0}$ is within P. Each iteration a new vector $\boldsymbol{y}$ is selected, such that $\boldsymbol{y}$ is as aligned with the current gradient of $F(\boldsymbol{x})$ as much as possible while remaining in the polytope. Then, a step size α is selected, which determines how much $\boldsymbol{y}$ will contribute to the new $\boldsymbol{x}$.

A second algorithm from Chekuri et al. (2011) is shown as Algorithm 8 here. We will refer to it as DOUBLE-LOCAL-OPT. It consists of first running LOCAL-OPT and then re-running it on a more restricted polytope. Chekuri et al. (2011) show that DOUBLE-LOCAL-OPT is a $\frac{1}{4}$-approximation. The next section covers a portion of that proof.

Chekuri et al. (2011) have a third algorithm as well, which is similar to LOCAL-OPT. The difference is that this other algorithm restricts its search to $P \cap [0, t]^\Omega$, for a fixed parameter t. Combining this algorithm with a simulated annealing procedure in which the "temperature" t is increased from 0 to 1, the authors are able to show an approximation factor of 0.325. That proof is beyond the scope of this survey, but some of its structure is similar to the proof of DOUBLE-LOCAL-OPT's approximation guarantee, which we derive in the next section.

Algorithm 7 LOCAL-OPT

Input: multilinear extension F, polytope P
$\boldsymbol{x} \leftarrow \mathbf{0}$
while not converged **do**
  $\boldsymbol{y} \leftarrow \arg\max_{\boldsymbol{y}' \in P} \nabla F(\boldsymbol{x})^\top \boldsymbol{y}'$
  $\alpha \leftarrow \arg\max_{\alpha' \in [0,1]} F(\alpha' \boldsymbol{x} + (1-\alpha')\boldsymbol{y})$
  $\boldsymbol{x} \leftarrow \alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}$
end while
return $\boldsymbol{x}$

Algorithm 8 DOUBLE-LOCAL-OPT

Input: multilinear extension F, polytope P
$\boldsymbol{x} \leftarrow$ LOCAL-OPT(F, P)
$\boldsymbol{y} \leftarrow$ LOCAL-OPT$(F, P \cap \{\boldsymbol{y}' \mid \boldsymbol{y}' \leq \mathbf{1} - \boldsymbol{x}\})$
return $\begin{cases} \boldsymbol{x} & : F(\boldsymbol{x}) > F(\boldsymbol{y}) \\ \boldsymbol{y} & : \text{otherwise} \end{cases}$

4.3 Selected Proofs

Chekuri et al. (2011) establish an approximation guarantee for DOUBLE-LOCAL-OPT. This section covers that guarantee and a portion of its proof.

Theorem 53. *Algorithm 8 (*DOUBLE-LOCAL-OPT*) is a $\frac{1}{4}$-approximation to Problem 49.*

The proof of this theorem relies on several lemmas. The first is as follows:

Lemma 54. *Given vectors $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^\Omega$, it is the case that: $F(\boldsymbol{x} \vee \boldsymbol{y}) + F(\boldsymbol{x} \wedge \boldsymbol{y}) - 2F(\boldsymbol{x}) \leq (\boldsymbol{y} - \boldsymbol{x}) \cdot \nabla F(\boldsymbol{x})$.*

Proof. First, notice that $0 \leq (\boldsymbol{x} \vee \boldsymbol{y}) - \boldsymbol{x}$ and that $(\boldsymbol{x} \wedge \boldsymbol{y}) - \boldsymbol{x} \leq 0$. Then Lemma 51 and Corollary 52, combined with the first-order definition of concavity imply:

$$F(\boldsymbol{x} \vee \boldsymbol{y}) - F(\boldsymbol{x}) \geq ((\boldsymbol{x} \vee \boldsymbol{y}) - \boldsymbol{x}) \cdot \nabla F(\boldsymbol{x}) \tag{4.7}$$
$$F(\boldsymbol{x} \wedge \boldsymbol{y}) - F(\boldsymbol{x}) \geq ((\boldsymbol{x} \wedge \boldsymbol{y}) - \boldsymbol{x}) \cdot \nabla F(\boldsymbol{x}) \,. \tag{4.8}$$

Summing these two equations we have:

$$F(\boldsymbol{x} \vee \boldsymbol{y}) + F(\boldsymbol{x} \wedge \boldsymbol{y}) - 2F(\boldsymbol{x}) \geq ((\boldsymbol{x} \vee \boldsymbol{y}) + (\boldsymbol{x} \wedge \boldsymbol{y}) - 2\boldsymbol{x}) \cdot \nabla F(\boldsymbol{x}) \,. \tag{4.9}$$

Applying the identity $(\boldsymbol{x} \vee \boldsymbol{y}) = (\boldsymbol{x} + \boldsymbol{y}) - (\boldsymbol{x} \wedge \boldsymbol{y})$ gives the desired result. □

A corollary of this first lemma clarifies its significance with respect to the optimization algorithms. Specifically, it considers the case where LOCAL-OPT has converged, which occurs when the $\boldsymbol{y}$-selection line of the algorithm sets $\boldsymbol{y} = \boldsymbol{x}$.

Corollary 55. *If $\boldsymbol{x}$ is a local optimum (i.e. $(\boldsymbol{y} - \boldsymbol{x}) \cdot \nabla F(\boldsymbol{x}) \leq 0$), then $2F(\boldsymbol{x}) \geq F(\boldsymbol{x} \vee \boldsymbol{y}) + F(\boldsymbol{x} \wedge \boldsymbol{y})$.*

The final lemma required concerns combining the results of DOUBLE-LOCAL-OPT's two LOCAL-OPT runs.

Lemma 56. *Let* OPT *represent the solution to Problem 49, the polytope-constrained multilinear maximization problem. If $\boldsymbol{x}$ is a local optimum in P and $\boldsymbol{y}$ is a local optimum in $P \cap \{\boldsymbol{y}' \mid \boldsymbol{y}' \leq \mathbf{1} - \boldsymbol{x}\}$, then* $\text{OPT} \leq 2F(\boldsymbol{x}) + 2F(\boldsymbol{y})$.

We won't cover the details of this lemma's proof here, but remark that it effectively shows Theorem 53 — that the average of the two LOCAL-OPT solutions is at least a $\frac{1}{4}$-approximation to Problem 49.

4.4 Contention Resolution Schemes

Problem 49, the polytope-constrained maximization problem, is defined with respect to the multilinear extension $F(\boldsymbol{x})$. But in practice, it is usually the case that a solution to $f(S)$ is needed. In the previous

chapter, for cardinality constraints, we mentioned that pipage rounding techniques would suffice to round $\boldsymbol{x}$. For general polytopes though, more complex mechanisms are necessary.

To this end, Chekuri et al. (2011) contribute a class of rounding algorithms that they call "contention resolution schemes" (CR schemes). Ultimately, given a continuous solution $\boldsymbol{x} \in P$, these schemes start from a random subset $R(\boldsymbol{x})$ that contains element i of Ω with probability x_i. Then, elements are removed from $R(\boldsymbol{x})$ until it is feasible according to the original constraints on S. The key is to design the removal algorithm in such a way that it is guaranteed element i appears in the final set with probability at least cx_i, for some sufficiently large $c > 0$.

Definition 57. (c-balanced CR scheme) *A CR scheme $\pi_{\boldsymbol{x}}$ is c-balanced if, for all $i \in support(\boldsymbol{x})$, it is the case that: $Pr[i \in \pi_{\boldsymbol{x}}(R(\boldsymbol{x})) \mid i \in R(\boldsymbol{x})] \geq c$.*

Notice that if f were a linear function, then a c-balanced scheme would convert $\boldsymbol{x}$ to a set S while keeping at least a c-fraction of $F(\boldsymbol{x})$'s value. That is, if the approximation factor for maximizing F were α, the overall approximation factor would be $c\alpha$. For more general submodular f, ensuring this requires proving that the applied CR scheme is montone.

Definition 58. (Montonicity of a CR scheme) *A CR scheme $\pi_{\boldsymbol{x}}$ is monotone if $Pr[i \in \pi_{\boldsymbol{x}}(A_1)] \leq Pr[i \in \pi_{\boldsymbol{x}}(A_2)]$ whenever $i \in A_1 \subseteq A_2$.*

In some cases, it is easier to obtain a large c-value for a scaled-down version of the polytope: $bP = \{b\boldsymbol{x} \mid \boldsymbol{x} \in P\}$, with $b \in [0, 1]$. Such schemes are then called (b, c)-balanced. Chekuri et al. (2011) further show that any (b, c)-balanced scheme can be converted into a bc-balanced scheme.

All of these schemes focus on the setting where the constraints on f are of the form $S \in \mathcal{I}$, for some family of feasible sets $\mathcal{I} \subseteq 2^{\Omega}$. The natural polytope relaxation in this case is the convex hull of $\mathcal{I}$: $P_{\mathcal{I}} = \text{conv}(e_I \mid I \in \mathcal{I})$, as was shown earlier for matroid and knapsack constraints. At a high level, one concrete method for finding a CR scheme in this setting proceeds by selecting a subset of $R(\boldsymbol{x})$ as follows:

- Let $\phi : 2^{\Omega} \rightarrow \mathcal{I}$ represent a mapping from possible $R(\boldsymbol{x})$ to feasible output sets $\mathcal{I}$. Let λ_{ϕ} be a distribution over all ϕ. Construct a linear programming problem where the objective value is c and the variables are the probabilities defining λ_{ϕ}. The program's constraints should ensure the c-balanced property.
- Solve this LP in the dual using the ellipsoid algorithm. Estimate the primal variables (probabilities of the λ_{ϕ} distribution) from the dual and draw a single ϕ according to these probabilities.
- Apply the mapping ϕ to $R(\boldsymbol{x})$ to obtain the final feasible set $S \in \mathcal{I}$.

Under this umbrella, one of the most interesting results Chekuri et al. (2011) obtain is an optimal CR scheme for a matroid constraint.

Theorem 59. *For all $b \in [0, 1]$, there exists a monotone $(b, \frac{1-e^{-b}}{b})$-balanced scheme for any matroid polytope. There is no (b, c)-balanced scheme with greater c for this problem.*

Notice that the choice $b = 1$ corresponds to $c = (1 - \frac{1}{e})$. Combining this with the $\frac{1}{4}$-approximation factor for the problem of maximizing F (Problem 49), the overall approximation guarantee is $\frac{1}{4}(1 - \frac{1}{e})$.

Chekuri et al. (2011) also provide CR schemes for knapsack constraints and several other common constraint types. What makes the CR framework truly compelling though is that a naïve combination of individual CR schemes produces another CR scheme with a surprisingly large c factor. More concretely, given h CR schemes, let the output produced by running each individual scheme, $\pi^1_{\boldsymbol{x}}, \ldots, \pi^h_{\boldsymbol{x}}$, be the h sets $I_1, \ldots, I_h$. For a combined output $\bigcap_{i=1}^{h} I_i$, Chekuri et al. (2011) provide the following guarantee:

Theorem 60. *Given a submodular maximization problem with constraints $\mathcal{I} = \bigcap_{i=1}^{h} \mathcal{I}_i$, suppose each corresponding relaxed polytope $P_{\mathcal{I}_i}$ has a monotone (b, c_i)-balanced CR scheme. Then the overall polytope $P_{\mathcal{I}} = \bigcap_{i=1}^{h} P_{\mathcal{I}_i}$ has a monotone $\left(b, \prod_{i=1}^{h} c_i\right)$-balanced CR scheme.*

4.5 Discussion

Devising submodular maximization algorithms for polytope constraints is an important avenue of research. As discussed earlier in this chapter, polytopes include significant classes of constraints, such as matroid and knapsack constraints. Substantial work has been done tackling these constraint classes individually (Lee et al., 2009; Vondrák, 2009; Gharan and Vondrák, 2011). But Chekuri et al. (2011)'s work represents a major step towards general-purpose algorithms for *all* down-monotone, solvable polytopes.

For future development of such general-purpose algorithms, continuous optimization schemes seem like the way forward. Combinatorial techniques applied in the discrete setting may work for a particular type of constraint, but aren't flexible enough to handle many different types of constraints. Chekuri et al. (2011)'s use of the multilinear extension illustrates the power of that particular continuous submodular variant. While the approximation factors associated with gradient-ascent-based algorithms such as DOUBLE-LOCAL-OPT (Algorithm 8) seem somewhat small, it's possible that, with a few additional assumptions on the nature of the constraints, these factors could be improved.

When working with continuous algorithms, there is the additional need for good rounding techniques. Again, Chekuri et al. (2011)'s work represents a major advancement in this area. Their contention resolution schemes are a departure from standard rounding algorithms in two main respects. First, they tend to rely on greater randomization; recall that the linear programming method for finding a CR scheme outputs a *distribution* over possible mappings from $R(\boldsymbol{x})$ to the final output set. Secondly, they are non-oblivious; the linear programming method for selecting a CR scheme depends on the particular value of $\boldsymbol{x}$ and so the final scheme is tailored to it. In short, CR schemes merit further investigation, including an exploration of their performance on real-world tasks.

We close this chapter by noting some hardness results that imply limits on approximation algorithms for general polytope constraints. Gharan and Vondrák (2011) show that a 0.478-approximation for a matroid independence constraint would require an exponential number of queries to an f-oracle. They also show an even stronger hardness result of 0.394 for a matroid base constraint when the matroid has two disjoint bases. These results limit the extent to which future work can improve on the guarantees of Chekuri et al. (2011), but do not imply that the Chekuri et al. (2011) results are tight.

Chapter 5

Online Setting

Up to this point, all of the algorithms we've considered apply only in the *offline* setting. Before choosing an item, every one of these algorithms performs some computation that requires knowledge of every item in $\Omega = \{1, \ldots, N\}$. For online settings, such as the secretary setting, this is not possible.

Definition 61. (Secretary setting) *Elements from Ω arrive one at a time, in a random sequence. The decision to accept or reject an item must be made before the subsequent item in the sequence arrives.*

The name "secretary" comes from an instance of this setting where the elements of Ω are applicants for a secretary job. The applicants are interviewed sequentially and the interviewer must decide directly after each interview whether to hire the applicant or dismiss them. In this basic application, there is an implicit cardinality constraint, $k = 1$. In general, there are practical applications with larger limits k and even limits based on more complex structures. For instance, recall the matroids from Section 1.2.3. If instead of a single secretary position there were ten different office jobs to fill, then the selection problem would be constrained by a partition matroid with ten parts. Further, suppose the office jobs contained an element of collaboration. Then relationships between the different applicants would be important, and overall office performance might be well-modeled as a non-monotone submodular function of the selected office workers. Most of the literature for the secretary setting focuses on a somewhat different application though — online auctions (Hajiaghayi et al., 2004; Kleinberg, 2005). For these auctions, a seller has c items available and bidders arrive and leave dynamically. Again, matroid constraints often arise.

To address secretary setting problems, alternative submodular maximization techniques are necessary. Gupta et al. (2010) provide several useful algorithms in this vein. The remainder of this chapter focuses on their algorithms and the associated guarantees. Before proceeding, we briefly note that Gupta et al. (2010) also give offline algorithms with approximation guarantees. Some of these are superseded by more recent results. For instance, in the case of their offline algorithm for the cardinality constrained setting, Buchbinder et al. (2014)'s RANDOM-GREEDY (Algorithm 4) achieves a better approximation factor without incurring any additional runtime cost. Other results, such as one associated with p-independence system constraints, may remain the best known approximation factors. We discuss the p-independence system result briefly in the conclusion, as it seems to suggest possible future work.

5.1 Selected Algorithms

The main algorithm highlighted in this section is Algorithm 9, SECRETARIES. Gupta et al. (2010) show that this algorithm is a constant-factor approximation to the following problem.

Problem 62. (Secretary setting cardinality-capped submodular maximization) *We are given a submodular function $f : 2^{\Omega} \to \mathbb{R}_+$ and a random ordering of Ω. Subject to the requirement that once an element $i \in \Omega$ has been examined a decision must be made to include i in S or reject i before examining the next element in the ordering, find:* $\max_{S \subseteq \Omega : |S| \leq k} f(S)$.

Gupta et al. (2010)'s algorithm for this problem is an amalgamation of various submodular maximization techniques, each executed with a certain probability. Half of the time only the DYNKIN algorithm (Algorithm 10) is executed. This is a classical algorithm frequently used in the secretary setting. For the $k = 1$ case, it is in fact the optimal way to select S. It proceeds by calculating the f-value for each of the first $\lfloor \frac{N}{e} \rfloor$ elements, keeping track of the max with the variable b. After that point, if it encounters an element with value greater than b, it returns that element as S. If no such element is encountered, it returns the empty set. In short, Algorithm 10 "ignores" the early elements to get a sense of what a good f-value is. It then returns a single "good" element if it encounters one in the remaining iterations.

If SECRETARIES doesn't execute Algorithm 10, then it instead begins by acquiring an estimate of OPT via a standard submodular maximization algorithm. That is, it feeds the first m elements, Ω_1, into an offline algorithm for the cardinality-capped submodular maximization problem (Problem 29). Gupta et al. (2010) propose one such offline algorithm, but, as mentioned earlier, Buchbinder et al. (2014)'s RANDOM-GREEDY (Algorithm 4) achieves a better approximation factor without incurring any additional runtime cost. Thus, in the specification of SECRETARIES here, we substitue in RANDOM-GREEDY.

Using the output of RANDOM-GREEDY as a proxy for OPT, the final step in SECRETARIES is to run one of three online "advice-taking" algorithms on the remainder of the data, Ω_2. The first algorithm is THRESHOLD, which simply accepts the first k elements of Ω that have value greater than τ. The second algorithm is THRESHOLD-RANDOM, which is exactly like THRESHOLD except it rejects with probability $\frac{1}{2}$ each element that THRESHOLD would have accepted. Essentially, THRESHOLD-RANDOM emulates the process of running random selection on the output of THRESHOLD. The third and final algorithm is another variation on THRESHOLD, THREHSOLD-OPPOSITE. This algorithm emulates the process of running THRESHOLD on $\Omega_2 \setminus S_1$. Note that these emulation algorithms are necessary because the secretary setting does not allow for two passes through the data; running THRESHOLD twice would not be acceptable.

Algorithm 9 SECRETARIES

Input: submodular function f, ground set Ω, limit k
with probability $\frac{1}{2}$ **do:**
  $S \leftarrow \text{DYNKIN}(f, \Omega)$
else do:
  $m \sim \text{Binomial}\left(N, \frac{1}{2}\right)$
  $\Omega_1 \leftarrow \{1, \ldots, m\}, \Omega_2 \leftarrow \{m+1, \ldots, N\}$
  $A \leftarrow \text{RANDOM-GREEDY}(f, \Omega_1, k)$
  $\tau \leftarrow f(A)/(7k)$
  $c \sim \text{Uniform}(0, 1)$
  if $c \leq \frac{1}{3}$ **then**
    $S_1 \leftarrow \text{THRESHOLD}(f, \Omega_2, k, \tau)$
  end if
  if $\frac{1}{3} < c \leq \frac{2}{3}$ **then**
    $S_1' \leftarrow \text{THRESHOLD-RANDOM}(f, \Omega_2, k, \tau)$
  end if
  if $\frac{2}{3} < c$ **then**
    $S_2 \leftarrow \text{THREHSOLD-OPPOSITE}(f, \Omega_2, k, \tau)$
  end if
return whichever of S, S_1, S_1', S_2 was created

Algorithm 10 DYNKIN

Input: submodular function f, ground set Ω
$S \leftarrow \emptyset, b \leftarrow -\infty$
for $i = 1$ **to** N **do**
  if $i < \frac{N}{e}$ **then**
    if $f(\{i\}) > b$ **then**
      $b \leftarrow f(\{i\})$
    end if
  else
    if $f(\{i\}) > b$ **then**
      $S \leftarrow \{i\}$
      return S
    end if
  end if
end for
return $\emptyset$

Algorithm 11 THRESHOLD

Input: submodular function f, ground set Ω, limit k, threshold τ
$S \leftarrow \emptyset$
for $i = 1$ **to** N **do**
  if $|S| = k$ **then**
    return S
  end if
  if $f(S \cup \{i\}) - f(S) > \tau$ **then**
    $S \leftarrow S \cup \{i\}$
  end if
end for
return S

Algorithm 12 THRESHOLD-RANDOM

Input: submodular function f, ground set Ω, limit k, threshold τ
$S \leftarrow \emptyset, c \leftarrow 0$
for $i = 1$ **to** N **do**
  if $c = k$ **then**
    return S
  end if
  if $f(S \cup \{i\}) - f(S) > \tau$ **then**
    $c \leftarrow c + 1$
    with probability $\frac{1}{2}$ **do:**
      $S \leftarrow S \cup \{i\}$
  end if
end for
return S

Algorithm 13 THREHSOLD-OPPOSITE

Input: submodular function f, ground set Ω, limit k, threshold τ
$S_1 \leftarrow \emptyset, S_2 \leftarrow \emptyset$
for $i = 1$ **to** N **do**
  if $|S_2| = k$ **then**
    return S_2
  end if
  if $f(S_1 \cup \{i\}) - f(S_1) > \tau$ and $|S_1| < k$ **then**
    $S_1 \leftarrow S_1 \cup \{i\}$
  else
    if $f(S_2 \cup \{i\}) - f(S_2) > \tau$ **then**
      $S_2 \leftarrow S_2 \cup \{i\}$
    end if
  end if
end for
return S_2

5.2 Selected Proofs

Theorem 63. SECRETARIES *(Algorithm 9) is an $O(1)$-approximation for the secretary setting cardinality-capped submodular maximization problem (Problem 62).*

Gupta et al. (2010)'s proof of SECRETARIES's approximation guarantee relies on several interesting, general lemmas. The first of these provides a bound on the value returned by the THRESHOLD algorithm.

Lemma 64. *Let C^* be a set with value $f(C^*) = \text{OPT}$. If* THRESHOLD *returns a set S of size $|S| < k$, this set is still guaranteed to have value at least $f(S \cup C^*) - |C^*|\tau$.*

Proof. Suppose not. That is, suppose $f(S) < f(S \cup C^*) - |C^*|\tau$. Then:

$$\tau < \frac{1}{|C^*|}(f(S \cup C^*) - f(S)) \leq \frac{1}{|C^*|}\left(\sum_{i \in C^*}(f(S \cup \{i\}) - f(S))\right) \tag{5.1}$$

where the last inequality follows by submodularity of f. Since the average marginal gain (over S) for elements in C^* is larger than τ, this implies at least one element $i \in C^*$ has marginal gain larger than τ. This element clearly isn't in S, or the marginal gain would be zero. Yet, by submodularity, at the point during THRESHOLD when i was considered, its marginal gain could only have been greater. This implies a contradiction, because a gain greater than τ would have placed i in S. □

The second lemma that the SECRETARIES proof relies on is not algorithm-specific, but rather of general interest. It is also used by Lee et al. (2009).

Lemma 65. *Given $S_1, C \subseteq \Omega$, define $C' = C \setminus S_1$ and $S_2 \subseteq \Omega \setminus S_1$. Then it is the case that: $f(C) \leq f(S_1 \cup C) + f(S_1 \cap C) + f(S_2 \cup C')$.*

Proof. Let $A_1 = C'$ and $B_1 = S_1 \cup C$. Then $A_1 \subseteq B_1$, and applying Definition 1 of submodularity yields:

$$f(S_1 \cup C \cup S_2) - f(S_1 \cup C) \leq f(C' \cup S_2) - f(C') \,. \tag{5.2}$$

Let $A_2 = C'$ and $B_2 = S_1 \cap C$. Then $A_2 \cap B_2 = \emptyset$ and $A_2 \cup B_2 = C$. Applying Definition 2 of submodularity yields:

$$f(C) + f(\emptyset) \leq f(C') + f(S_1 \cap C) \,. \tag{5.3}$$

Summing these these two inequalities:

$$f(C) + f(\emptyset) + f(S_1 \cup C \cup S_2) - f(S_1 \cup C) \leq f(C' \cup S_2) + f(S_1 \cap C) \tag{5.4}$$

$$f(C) + f(\emptyset) + f(S_1 \cup C \cup S_2) \leq f(S_1 \cup C) + f(C' \cup S_2) + f(S_1 \cap C) \tag{5.5}$$

$$f(C) \leq f(S_1 \cup C) + f(C' \cup S_2) + f(S_1 \cap C) \tag{5.6}$$

where the last line above follows from non-negativity of f. □

The third lemma that the SECRETARIES proof relies on establishes a safe setting for THRESHOLD's τ.

Lemma 66. *Given the true value of* OPT *and letting $\Omega_2 = \Omega$, executing the second (non-*DYNKIN*) branch of* SECRETARIES *with the setting $\tau = \frac{\text{OPT}}{7k}$ results in a set S with expected f-value of at least* OPT$/21$.

Proof. We begin by lower-bounding the sum of the outputs of the three possible code branches, $f(S_1) + f(S_1') + f(S_2)$. We show the sum is at least τk.

- Case 1 — assume $|S_1| = k$ or $|S_2| = k$: S_1 and S_2 are selected using variants of the THRESHOLD algorithm. These algorithms only add an item if it has value greater than τ, so the result is immediate.
- Case 2 — assume $4\tau k \leq f(S_1 \cap C^*)$: The algorithm for finding S_1' emulates independently sampling elements from S_1 with probability $\frac{1}{2}$. Theorem 2.1 from Feige et al. (2007) characterizes such a random selection algorithm as a $\frac{1}{4}$-approximation. Thus, we have $f(S_1') \geq \tau k$.
- Case 3 — assume conditions from cases 1 and 2 are both violated:

$$f(S_1) + f(S_2) \geq (f(S_1 \cup C^*) - |C^*|\tau) + (f(S_2 \cup (C^* \setminus S_1)) - |C^*|\tau) \tag{5.7}$$

Applying Lemma 64.

$$\geq (f(S_1 \cup C^*) - k\tau) + (f(S_2 \cup (C^* \setminus S_1)) - k\tau) \tag{5.8}$$

Using the fact that $|C^*| \leq k$.

$$\geq (f(S_1 \cup C^*) - k\tau) + (f(S_2 \cup (C^* \setminus S_1)) - k\tau) + (f(S_1 \cap C^*) - 4k\tau) \tag{5.9}$$

Using the fact that case 2's assumption is violated.

$$\geq f(C^*) - 6k\tau \tag{5.10}$$

Applying Lemma 65.

$$= \text{OPT} - 6k\tau = 7k\tau - 6k\tau = k\tau \,. \tag{5.11}$$

Applying the definition of τ.

Having shown $f(S_1) + f(S_1') + f(S_2) \geq \tau k$, note that the second branch of SECRETARIES returns each one of S_1, S_1', and S_2 with probability $\frac{1}{3}$. Thus, dividing by 3 and substituting in the assumed setting for τ gives a lower bound of $\frac{\tau k}{3} = \frac{\text{OPT}}{21}$. □

The remainder of the proof for the SECRETARIES algorithm essentially shows that removing m elements to estimate OPT (and then using the subsequent estimate instead of the true OPT) still results in a constant-factor approximation. Combining this with the DYNKIN algorithm's guarantees, Gupta et al. (2010) ultimately show the expected f-value of the returned set is at least OPT/1417. While this isn't the most inspiring constant factor, it is nonetheless a starting point for future work.

5.3 Discussion

The online setting is an important and practical realm for submodular maximization. One unifying characteristic of all work in the secretary setting is that the algorithms use a fraction of the early input to get a sense of what a "good" f-value is, then select from the remainder of the input stream using the "good" value to guide the selection. There has been some recent research exploring *monotone* submodular maximization in the secretary setting (Feldman et al., 2011a; Ma et al., 2013), but Gupta et al. (2010)'s work remains one of the only publications to consider *non-monotone* functions.

In addition to the constant-factor guarantee demonstrated above, Gupta et al. (2010) provide guarantees for matroid constraints as well. For the partition matroid, they again show an $O(1)$-approximation, and for a general rank-k matroid an $O(\frac{1}{\log k})$-approximation. The work of Bateni et al. (2010), done concurrently with and independently of Gupta et al. (2010)'s work, establishes similar guarantees. For the *monotone* cardinality-capped case they offer an algorithm with an approximation factor of $\frac{1-1/e}{11}$. This algorithm essentially segments the input stream into k equal-size chunks and runs DYNKIN on each chunk. Using that algorithm as a subroutine in a larger algorithm, Bateni et al. (2010) are also able to prove a constant-factor approximation for *non-monotone* functions. Their algorithm is a $\frac{1}{8e^2}$-approximation to Problem 62. Additionally, Bateni et al. (2010) address the case of knapsack constraints. For k knapsacks, they find an $O(\frac{1}{k})$-approximation algorithm.

In terms of the tightness of the current approximation guarantees, not much is known. For the rank-k matroid case, Babaioff et al. (2007) consider the simpler setting where f is linear, and prove an $O(\frac{1}{\log k})$-approximation factor. This matches Gupta et al. (2010)'s result for submodular f, but leaves open the question of whether either approximation is tight.

Chapter 6

Conclusion

In this survey we have considered the problem of non-monotone submodular maximization across a variety of settings. Chapter 2 addressed the unconstrained setting, for which Buchbinder et al. (2012) provide a tight $\frac{1}{2}$-approximation algorithm. Chapter 3 addressed the cardinality-constrained setting. Buchbinder et al. (2014)'s algorithms in this setting are tight for $k = \frac{N}{2}$, but possibly sub-optimal elsewhere. Chapter 4 addressed the setting in which the convex hull of the feasible sets forms a down-monotone, solvable polytope. Chekuri et al. (2011) tackle this setting by relaxing to the continuous realm and applying novel rounding schemes. Finally, Chapter 5 addressed the online setting. Gupta et al. (2010) provide a constant-factor guarantee for the cardinality-constrained problems in this setting, along with other several other guarantees for matroid constraints.

Not all of the approximations discussed are known to be tight, so there is most likely room for improvement. A few directions here seem promising. First, notice that many of the algorithms we saw include a complement operation. In Buchbinder et al. (2012) and Buchbinder et al. (2014), this is implicitly done by starting from $f(\emptyset)$ and $f(\Omega)$. Effectively, this is like optimizing both $f(S)$ and $\bar{f}(S) = f(\Omega \setminus S)$. In Chekuri et al. (2011), a similar pattern appears. DOUBLE-LOCAL-OPT (Algorithm 8) first optimizes over the polytope P to find $\boldsymbol{x}$, then optimizes again over $P \cap \{\boldsymbol{y} \mid \boldsymbol{y} \leq \mathbf{1} - \boldsymbol{x}\}$. Gupta et al. (2010)'s algorithms include a implicit complement as well, with THREHSOLD-OPPOSITE (Algorithm 13) avoiding all elements that would have gone into S_1. Complementing f is one means of producing an alternative function to optimize, but there are also other simple operations that preserve submodularity. For instance, restriction: for submodular f, the restriction to $A \subseteq \Omega$ is also a submodular function: $g(S) = f(S \cap A)$. Perhaps there are interesting choices of restricting set A that could yield new, useful submodular maximization algorithms.

A more concrete opportunity for improvement presents itself in Gupta et al. (2010)'s (offline) algorithm for handling p-independence system constraints. We did not cover the algorithm in this survey, but it relies, in part, on a slightly modified version of NEMHAUSER-WOLSEY (Algorithm 1). Specifically, instead of blindly selecting the item $\{j\}$ that achieves maximum marginal gain, it first checks to make sure that $S \cup \{j\}$ is an independent set. The same check could instead be added to RANDOMIZED-USM (Algorithm 3), possibly resulting in an algorithm with better approximation guarantees.

In addition to improving the general-purpose algorithms, it is of practical interest to adapt them for use with individual submodular functions. For instance, recall the document summarization task described in Section 1.3.2 and its log determinant objective (Problem 19). The multilinear extension F for log determinant cannot be efficiently computed; we have to resort to sampling to estimate its value. Only a polynomial number of samples is necessary, but this process can still be prohibitively slow for real-world tasks. Yet, by making a slight modification to the multilinear extension and exploiting properties particular to the individual submodular function under consideration (log determinant), an efficiently computable version emerges (Gillenwater et al., 2012). There may very well be other submodular functions for which similar minor changes to the general-purpose algorithms result in more practical, specialized algorithms.

In summary, the current body of work on non-monotone submodular maximization has laid strong foundations, and there remains ample room for future development.

Acknowledgements

Many thanks to my WPE-II committee members—Aaron Roth, Michael Kearns, and Sanjeev Khanna—for their time and their astute questions. I would also like to thank my advisor, Ben Taskar: first, for the use of his cooler, without which my office food would doubtless have been eaten by the office mice long before I could have finished this report; and second, more generally, for his enthusiasm and his dedication, the memory of which continues to inspire and uplift. Finally, thanks to Alex Kulesza, for sending me the email that Ben would have sent if he had seen this report.

Bibliography

Ageev, A. and Sviridenko, M. (1999). An 0.828-Approximation Algorithm for the Uncapacitated Facility Location Problem. *Discrete Applied Mathematics*, 93:149–156.

Babaioff, M., Immorlica, N., and Kleinberg, R. (2007). Matroids, Secretary Problems, and Online Mechanisms. In *SODA*.

Bach, F. (2013). Learning with Submodular Functions: A Convex Optimization Perspective. Technical Report HAL 00645271-v2.

Bateni, M., Hajiaghayi, M., and Zadimoghaddam, M. (2010). Submodular Secretary Problem and Extensions. Technical Report MIT-CSAIL-TR-2010-002.

Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. (2012). A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization. In *FOCS*.

Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. (2014). Submodular Maximization with Cardinality Constraints. In *SODA*.

Calinescu, G., Chekuri, C., Pál, M., and Vondrák, J. (2007). Maximizing a Submodular Set Function Subject to a Matroid Constraint. In *IPCO*.

Calinescu, G., Chekuri, C., Pál, M., and Vondrák, J. (2011). Maximizing a Monotone Submodular Function Subject to a Matroid Constraint. *SICOMP*, 40:1740–1766.

Chekuri, C., Vondrák, J., and Zenklusen, R. (2011). Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes. In *STOC*.

Dobzinski, S. and Vondrák, J. (2012). From Query Complexity to Computational Complexity. In *STOC*.

Dughmi, S., Roughgarden, T., and Sundararajan, M. (2009). Revenue Submodularity. In *Electronic Commerce*.

Feige, U., Mirrokni, V., and Vondrák, J. (2007). Maximizing Non-Monotone Submodular Functions. In *FOCS*.

Feldman, M., Naor, J., and Schwartz, R. (2011a). Improved Competitive Ratios for Submodular Secretary Problems. In *Workshop on Approximation Algorithms for Combinatorial Optimization Problems*.

Feldman, M., Naor, J., and Schwartz, R. (2011b). A Unified Continuous Greedy Algorithm for Submodular Maximization. In *FOCS*.

Frank, M. and Wolfe, P. (1956). An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly*, 3:95–110.

Frieze, A. (1974). A cost function property for plant location problems. *Mathematical Programming*, 7:245–248.

Gharan, S. and Vondrák, J. (2011). Submodular Maximization by Simulated Annealing. In *SODA*.

Gillenwater, J., Kulesza, A., and Taskar, B. (2012). Near-Optimal MAP Inference for Determinantal Point Processes. In *NIPS*.

Goemans, M. and Williamson, D. (1995). Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the ACM*, 42:1115–1145.

Gupta, A., Roth, A., Schoenebeck, G., and Talwar, K. (2010). Constrained Nonmonotone Submodular Maximization: Offline and Secretary Algorithms. In *WINE*.

Hajiaghayi, M., Kleinberg, R., and Parkes, D. (2004). Adaptive Limited-Supply Online Auctions. In *Electronic Commerce*.

Hartline, J., Mirrokni, V., and Sundararajan, M. (2008). Optimal Marketing Strategies over Social Networks. In *WWW*.

Khuller, S., Moss, A., and Naor, J. (1999). The Budgeted Maximum Coverage Problem. *Information Processing Letters*, 70:39–45.

Kim, G., Xing, E., Fei-Fei, L., and Kanade, T. (2011). Distributed Cosegmentation via Submodular Optimization on Anisotropic Diffusion. In *ICCV*.

Kleinberg, R. (2005). A Multiple-Choice Secretary Algorithm with Applications to Online Auctions. In *SODA*.

Krause, A. and Guestrin, C. (2005). Near-Optimal Non-Myopic Value of Information in Graphical Models. In *UAI*.

Krause, A., Singh, A., and Guestrin, C. (2008). Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms, and Empirical Studies. *JMLR*, 9:235–284.

Kulesza, A. and Taskar, B. (2012). Determinantal Point Processes for Machine Learning. *Foundations and Trends in Machine Learning*.

Lee, J., Mirrokni, V., Nagarajan, V., and Sviridenko, M. (2009). Maximizing Non-Monotone Submodular Functions Under Matroid and Knapsack Constraints. In *STOC*.

Lin, H. and Bilmes, J. (2012). Learning Mixtures of Submodular Shells with Application to Document Summarization. In *UAI*.

Ma, T., Tang, B., and Wang, Y. (2013). The Simulated Greedy Algorithm for Several Submodular Secretary Problems. In *STACS*.

Mossel, E. and Roch, S. (2007). On the Submodularity of Influence in Social Networks. In *STOC*.

Nemhauser, G., Wolsey, L., and Fisher, M. (1978). An Analysis of the Approximations for Maximizing Submodular Set Functions - I. *Mathematical Programming*, 14:265–294.

Schulz, A. and Uhan, N. (2013). Approximating the Least Core Value and Least Core of Cooperative Games with Supermodular Costs. *Discrete Optimization*, 10:163–180.

Valiant, L. and Vazirani, V. (1986). NP is as Easy as Detecting Unique Solutions. *Theoretical Computer Science*, 47:85–93.

Vondrák, J. (2008). Optimal Approximation for the Submodular Welfare Problem in the Value Oracle Model. In *STOC*.

Vondrák, J. (2009). Symmetry and Approximability of Submodular Maximization Problems. In *FOCS*.

Watts, D. and Strogatz, S. (1998). Collective Dynamics of "Small-World" Networks. *Nature*, 393:440–442.