Mini-Project 1
A Survey on Optimization of Submodular Functions
CS 7001

Payam Siyari

Under Supervision of: Prof. Bistra Dilkina

December 3, 2014

Contents

1 Introduction

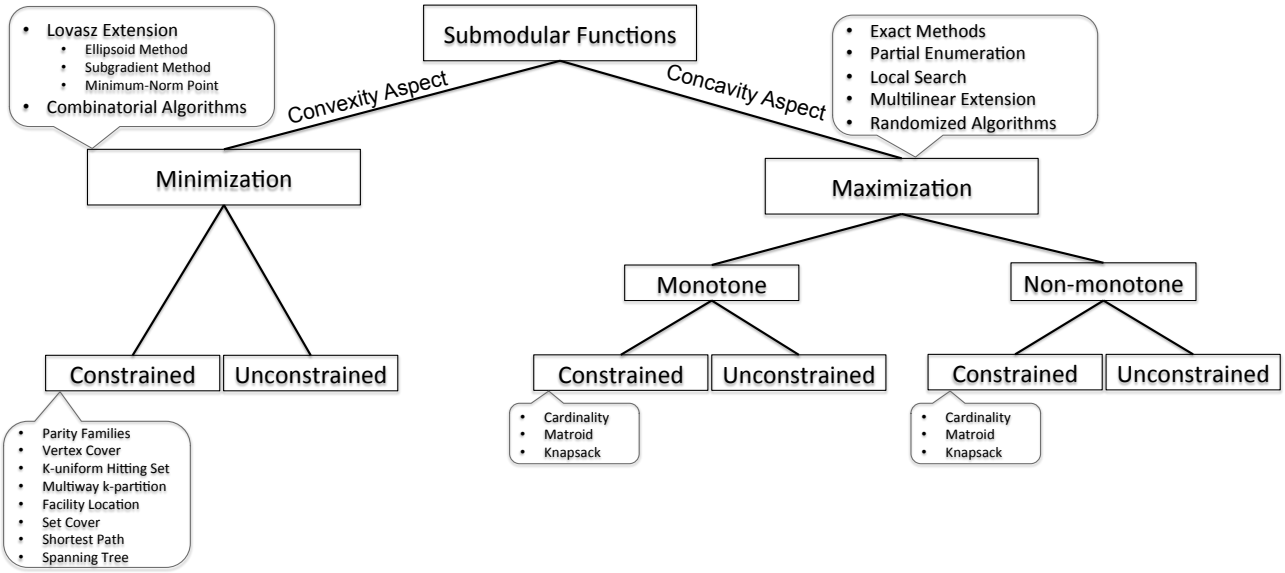
Of all the properties of set functions, submodularity has been one of the most attractive ones for researchers in recent years, if not the most. Submodularity has many applications in computer science, e.g. network inference [1], document summarization and MAP inference [2], and many surveys have been conducted to provide a general view of the vast literature about this property, e.g. [3] and [4]. Each of these surveys cover a subset of the results regarding the optimization of submodular function and in the mean time, they have many overlaps. In this report, we try to aggregate all of these surveys in one document to provide a consensus between all these surveys on the submodular function optimization.

After some preliminary notation and properties of submodular functions, in section 3, we cover the recent results on the minimization of submodular functions. Afterwards, in section 4 we cover the topics related to maximization of these functions.

Surprisingly, submodular set functions have common aspects with the analysis of both convex and concave functions. Hence firstly in section 3 and 4, we will show how these common aspects lead to the introduction of a set of techniques for optimization of these functions.

As in every optimization problem, each of the sections will be covering results for both unconstrained and constrained settings. However prior to this, a more generalized categorization of results in submodular function maximization is proposed which is whether the function is monotone or not. Of course, techniques for non-monotone functions can be applied to monotone functions, too. However, many interesting and tighter-bounded results have been proposed for monotone submodular maximization.

Figure below shows a summary of what will be presented in this report:



A diagram summary of this report

2 Preliminaries

2.1 Submodular Functions

Suppose V is a set, which is called the *ground set* in the literature of submodularity. A submodular function $f : 2^V \to \Re$ will satisfy the following:

> Suppose $A, B \subseteq V$ where $A \subseteq B$. If for every $x \in V \backslash B$, we have $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$, then f is a submodular function.

Another definition of a *submodular* function (which is a consequent of the previous definition) is:

> Suppose $A, B \subseteq V$. If we have $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, then f is a submodular function.

Similarly, we can define *supermodular* functions which can be simply considered as the negation of a submodular function. Also, if a function is both submodular and supermodular (changing the inequality in the definition of submodularity/supermodularity to equality), the function will be called a *modular* function.

2.2 Properties of Submodular Functions

2.2.1 Monotonocity

A submodular function $f : 2^V \to \Re$ is monotone iff $\forall A \subseteq B \subseteq |V|$, we have $f(A) \leq f(B)$.

2.2.2 Non-negativity

A set function $f : 2^V \to \Re$ is nonnegative iff $\forall S \subseteq V$, we have $f(S) \geq 0$.

2.2.3 Normalizaion

A set function $f : 2^V \to \Re$ is normalized iff $f(\phi) = 0$.

3 Submodular Function Minimization

3.1 Convexity Aspect of Submodular Functions

Depending on the problem of interest and one's interpretation, submodular set functions depict similar properties to convex and concave functions.

Lovasz [5], showed that submodularity of a set function f is equivalent to convexity of a continuous function f_L. In fact, this continuous function f_L is formed from an extension to the original set function f in a way that the minimum of f_L can be converted back to the minimizer of f. Hence, to minimize f, we can simply (and efficiently) minimize f_L and find the minimizer of f.

3.1.1 Convex Extension of Submodular Functions

It is almost straightforward that any set function $f : 2^V \rightarrow \Re$ can be represented as a function on binary vectors $f : \{0,1\}^{|V|} \rightarrow \Re$ where each entry of a vector represents the presence of its corresponding entry in a set. We show the corresponding binary vector for the set S as 1_S.

Lovasz Extension: Suppose $f : \{0,1\}^{|V|} \rightarrow \Re$. Lovasz extension of f will be $f_L : [0,1]^{|V|} \rightarrow \Re$ where:

$$f_L(x) = \sum_{i=0}^{|V|} \alpha_i f(S_i) \tag{3.1}$$

where $x = \sum_{i=0}^{|V|} \alpha_i 1_{S_i}$, $\sum_{i=0}^{|V|} \alpha_i = 1$, $\forall\ i \quad \alpha_i \geq 0$ and $\phi = S_0 \subset S_1 \subset ... \subset S_{|V|}$ There are also a number of other equivalent definitions for Lovasz extension [6]. What's important is that Lovasz extension f_L is a convex function. It is shown that Lovasz extension itself can be efficiently evaluated [7].

3.2 Example Problems

The examples below are mentioned in [8].

3.2.1 Recovery under Structured Sparsity and Dictionary Selection

Generally the sparse recovery problem can be specified as follows:

$$min_x \|y - Mx\|^2 + \lambda\Omega(x)$$

where y is the observed signal and columns of the matrix M represent the bases that we use to represent y on. Hence, x is simply a subset selection from these set of bases. λ and $\Omega(x)$ are the regularization parameter and regularization term, respectively, in order to induce sparsity constraint on x which means that the goal is to pick as few bases as possible. A first choice for $\Omega(x)$ will be $\|x\|_0$ or simply the cardinality of the selected subset of bases. However, with this choice the problem becomes NP-Hard and a next choice is $\|x\|_1$ as a convex envelop of $\|x\|_0$. However, the columns that are picked are not always arbitrary and depend on the chosen basis. For example, when the choice is wavelet bases, the selected columns must lie in a subtree of a wavelet tree. In this case, both functions $\|x\|_0$ and $\|x\|_1$ do not consider such constraint. A better choice will be a set function that is defined as follows:

$$\Omega(x) = |\cup_{s \in 1_x} ancestors(s)|$$

where s's are the nodes corresponding to nonzero values of 1_x and $ancestors(s)$ are all the ancestors of s in the wavelet tree. It can be seen that the function above is a submodular function and similar to what is done for $\|x\|_0$, we consider a convex envelop of this function (its Lovasz extension) in the final optimization problem. The optimization procedure for this problem is similar minimum-norm point algorithm.

In addition to the recovery problem, [9] shows that the selection of the dictionary columns and the sparse representation of signals can be presented as a joint combinatorial optimization problem. The authors show that under incoherence condition, this objective function will be approximately submodular. A set function f is approximately submodular, iff supposing $A, B \subseteq V$ where $A \subseteq B$, then for every $x \in V \backslash B$, we have $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B) - \epsilon$ where ϵ is a constant.

3.2.2 MAP Inference

Consider the problem of binary image segmentation where we want to separate a foreground object from a background. In fact, what we are trying to do is to label each pixel of the image as background or foreground, hence, assigning a binary label. Probabilistically, we want to maximize the posterior probability of labels w.r.t. to the pixel values. It can be shown that this posterior probability can be written as:

$$P(x|z) \propto exp(-E(x;z)) \tag{3.2}$$

where x is the label value vector, z is the pixel value vector and E is an Energy function. Hence, maximizing this posterior is equivalent to minimizing the energy function E. As E is a function on binary vectors (labels are binary), it is equivalent to a set function. Hence, if this equivalent set function is submodular, the minimizer of this submodular function will also have an equivalent that minimizes the energy function which maximizes the original posterior probability. Of course, there are cases that faster algorithms can be developed for the submodular minimization programs (which is needed for example with an image of a million pixels), such as when the energy function is of the form:

$$E(x;z) = \sum_i E_i(x_i) + \sum_{i,j} E_{i,j}(x_i, x_j) \tag{3.3}$$

and satisfies:

$$E_{i,j}(0,1) + E_{i,j}(1,0) \geq E_{i,j}(1,1) + E_{i,j}(0,0) \tag{3.4}$$

, the corresponding set function will be a graph-cut function. Hence, the MAP inference will actually be a minimum cut algorithm and for this special problem, it will be done in linear time to $|V|$.

3.3 Unconstrained Setting

The problem being solved is as follows:

$$min_{S \subseteq V}\ f(S) \tag{3.5}$$

where f is a submodular function.

3.3.1 Algorithms Based on Convex Extensions

1. **Ellipsoid Algorithm [10, 11]:** Earliest polynomial time algorithm for minimizing the convex extension of a submodular function was presented in [10]. The strongly polynomial time version was subsequently introduced by the same authors in [11]. Both methods were based on ellipsoid method, which was first used in [12] for polynomial time algorithms in linear programming.

2. **Subgradient Method [13]:** Although ellipsoid methods have polynomial time complexity, they are not efficient in practice and become intractable for larger problems. This is because of the fact that Lovasz extension is not a smooth function and subgradient descent algorithm converges slowly ($O(\frac{1}{\epsilon^2})$ steps to achieve ϵ accuracy). Authors in [13], develop an algorithm for minimizing *decomposable submodular functions* (submodular functions which can be represented as sums of concave functions applied to modular functions) with large number of variables. They show that their algorithm needs $O(\frac{D}{\epsilon})$ steps to achieve ϵ accuracy.

3. **Minimum-Norm Point Algorithm [14]:** The authors in [14] use the algorithm presented in [15] in submodular function minimization. This algorithm finds the minimum-norm point in the convex hull of a set of points in $\Re^n$. Although the authors leave it open to determine the complexity of using the mentioned in submodular function minimization, they experimentally show that it is strongly polynomial and orders of magnitude faster than the combinatorial algorithms.

3.3.2 Combinatorial Algorithms

1. Combinatorial strongly polynomial time algorithms for submodular function minimization were first presented in [16, 17]. [17] as an extension of [16], computes the minimizer in a pseudo-polynomial time and is based on network flow algorithms.

2. Building up on the works in [16, 17], [18] (by employing a scaling scheme)[19] (by lexicographic augmentation framework) and [20, 21] (by a push/relabel algorithm using ideas in [19]) were presented to improve the running time of the previous algorithms.

3. Methods in [22] and [23] are currently state-of-the-art weekly and strongly polynomial algorithms for submodular function minimization with time complexities $O(|V|^4T + |V|^5 \log M)$ and $O(|V|^6 + |V|^5T)$, respectively where T is the number of evaluations of the submodular function and M is the maximum value of the submodular function. Further, by combining the approaches in [22] and [23], the method presented in [24] almost has the same weakly and strongly polynomial complexity.

Fig. 1 (from [25]) shows the hierarchy of the improvements to unconstrained submodular function minimization algorithms.

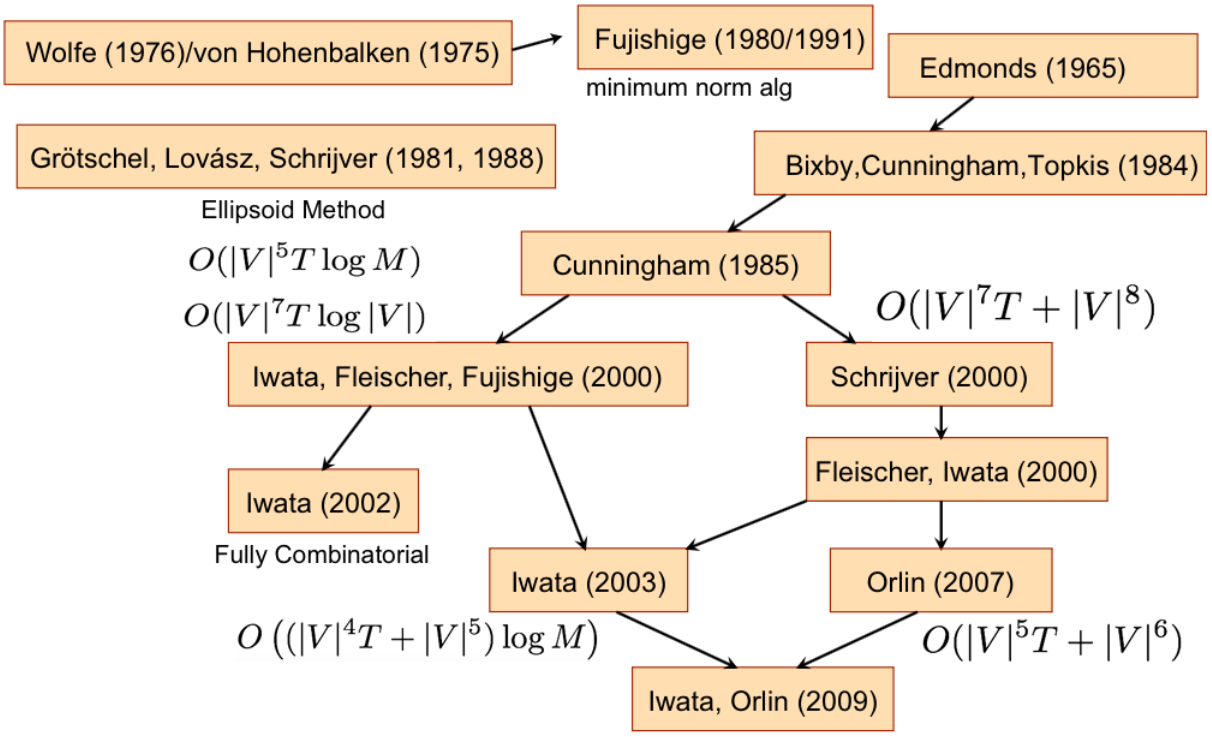


Figure 1: The hierarchy of the improvements to unconstrained submodular function minimization algorithms[25]

3.4 Constrained Setting

It has been shown that submodular functions can be minimized over a number of families of sets. These families include:

1. Lattices, odd/even sets, T-odd sets, T-even sets [10]
2. Parity families [26]
3. Inclusion closed families (for symmetric submodular functions) [27]

Unfortunately, adding a simple covering constraint makes the problem of submodular function minimization an NP-Hard problem. Hence, researchers have switched to approximation algorithms to solve these problems. We do not go in too much detail about these problems and borrow the table from [28] in this section for an overview of the literature in this area.

Table 1: Overview of Constrained Submodular Function Minimization [28]

Ref.	Constraint	Approximation	Hardness	Algorithmic Technique
[26]	Parity Families	1	1	Combinatorial
[29]	Vertex Cover	2	2	Lovasz Ext.
[30]	k-uniform Hitting Set	k	k	Lovasz Ext.
[31]	Multiway k-partition	2-2/k	2-2/k	Lovasz Ext.
[32]	Facility Location	$\log \lvert V \rvert$	$\log \lvert V \rvert$	Combinatorial
[24]	Set Cover	$\lvert V \rvert$	$\lvert V \rvert / \log^2 \lvert V \rvert$	Trivial
[33]	Shortest Path	$O(\lvert V \rvert^{2/3})$	$\Omega(\lvert V \rvert^{2/3})$	Combinatorial
[33]	Spanning Tree	$O(\lvert V \rvert)$	$\Omega(\lvert V \rvert)$	Combinatorial

3.5 Possible Future Work

Despite the huge literature, there are still a number of interesting problems in the field of submodular function minimization:

1. **Large-scale submodular function minimization:** Although minimum-norm point algorithm empirically works orders of magnitude faster than state-of-the-art combinatorial methods, developing methods that have provable guarantees for efficient minimization of submodular functions in presence of large number of data is of high interest.

2. **Approximation of constrained submodular minimization problems:** As mentioned, recent results show that submodular function minimization problems with almost all constraints are hard to approximate within a constant factor. However, still there exists problems in game theory and machine learning that proposing new constraints and new approximation bounds on submodular function minimization are of interest. For one such works, please refer to [34].

4 Submodular Function Maximization

4.1 Concavity Aspect of Submodular Functions

Submodular functions can be naturally considered as the discrete analogy of concave functions. We can view this analogy as follows:

$$\begin{cases} \textit{Submodularity}: & A \subseteq B, s \notin B \Rightarrow f(A \cup s) - f(A) \geq f(B \cup s) - f(B) \\ \textit{Concavity}: & a \leq b, s < 0 \Rightarrow f(a+s) - f(a) \geq f(b+s) - f(b) \end{cases} \tag{4.1}$$

4.2 Example Problems

There are many problems that can be formulated as a submodular function maximization. Here, we introduce five of such problems from [35].

4.2.1 Max-cut

Given a graph $G = (V, E)$, find the subset S_{max} that maximizes $f : 2^V \rightarrow 2^E$ where $\forall S \ f(S) = \{e \in E | e = (u, v) s.t. u \in S \ \& \ v \in V \backslash S\}$

4.2.2 Max-k-coverage

Select at most k sets from $\{A_1, ..., A_N\}$ such that their union has the largest cardinality, i.e. $max_{S:|S| \leq k} | \cup_{i \in S} A_i|$.

4.2.3 Document Summarization

The goal in document summarization is to select a subset S of the sentences from a a set of documents that summarizes them, usually subject to the constraint $|S| \leq k$. Maximum coverage is usually the most common criteria for choosing the submodular function in this problem, e.g. the max-k-coverage.

4.2.4 Network Inference

The problem of network inference is generally defined as the identification of underlying network topology by monitoring the influence data over this network. [1] formulates the problem of network inference under the assumption of independent cascade model [36]. Specifically, the authors define a likelihood for a set of information cascades over the network and show that this likelihood is a monotone submodular function. Consequently, they use the classic greedy approach for maximizing a monotone submodular function in order to predict the underlying network edges.

4.3 Exact Maximization

Contrary to what is true for unconstrained submodular function minimization, that is they can be solved in polynomial time, unconstrained maximization of a submodular function is NP-Hard in general. That means we assume the submodular function will have positive and negative values. In this case, it can be shown that getting $O(n^{1-\epsilon})$ approximation will be NP-Hard [3].

Despite the facts above, there have been a number of attempts, mostly based on mixed integer programming, for exact maximization of submodular functions. It is shown that submodular functions do not have a compact description in terms of integer programming. In [37], the authors give a constraint generation algorithm and a branch-and-bound algorithm that uses linear programming relaxations in order to find the maximizer. Other works in this area include [38, 39]. The important observation about all of these techniques is that due to the NP-Hardness of the unconstrained maximization problem, these algorithms are all exponential in the worst case and hence, only useful for small-sized problems.

Note that, all the algorithms that we consider in the following sections will assume a non-negative and normalized submodular function being maximized.

4.4 Monotonecity vs. Non-monotonecity

Although we can categorize the types of submodular maximization problems as constrained and unconstrained, the more challenging assumption in submodular function maximization is whether the submodular function is monotone or not. Hence, we categorize submodular function maximization

problems under two categories of monotone and non-monotone objectives. Before going into each of these categories, we will define different types of constraints applied to submodular maximization methods.

4.5 Constraints

4.5.1 Matroids

[35] A *matroid* is a pair $(V, \mathcal{I})$ such that V is a finite set and $\mathcal{I} \in 2^V$ is a set of sets S_i. Sets in $\mathcal{I}$ are *independent* in the sense that:

1. if $S_j \subseteq S_i \subseteq V$ and $B \in \mathcal{I}$, then $A \in \mathcal{I}$.
2. If $A, B \in \mathcal{I}$ and $|B| > |A|$, then there is $e \in B \backslash A$ such that $A \cup \{a\} \in \mathcal{I}$.

A simple example of a matroid is the matroid corresponding to the set of independent rows/columns in a matrix.

List below shows a number of types of matroids defined throughout the literature [35]:

1. **Uniform Matroid:** A matroid in which $\mathcal{I}$ only consists of sets $S \subseteq V$ of size at most k.
2. **Linear Matroid:** A matroid in which $\mathcal{I}$ consists the sets of linearly independent columns (or rows) in a matrix M.
3. **Graphic Matroid:** A matroid in which $\mathcal{I}$ consists the sets of edges in a graph $G = (V, E)$ that do not contain a cycle.
4. **Matching Matroid:** A matroid in which $\mathcal{I}$ consists the sets of vertices in a graph $G = (V, E)$ that can be covered by a matching.
5. **Partition Matroid:** A matroid in which $\mathcal{I}$ consists the sets of elements satisfying a partition constraint on V. Specifically, if V is partitioned to $P_1, ..., P_k$ and we have the numbers $a_1, ..., a_k$ associated with each partition. In a partition matroid, each set in I only consists less than or equal to a_i elements of partition P_i for all $1 \leq i \leq k$

4.5.2 Independence Matroid Polytope

The polytope $P_{\mathcal{I}} = ConvexHull(1_S | S \in \mathcal{I})$ for a matroid $(V, \mathcal{I})$ is called an independence matroid polytope.

4.5.3 Knapsack Constraints

A *knapsack* constraint is of the form $\sum_{i=1}^{|V|} a_i x_i \leq b$ for $a_i, x_i, b \geq 0$.

4.5.4 Knapsack Polytope

For a knapsack constraint $\sum_{i=1}^{|V|} a_i x_i \leq b$, the polytope $ConvexHull(x | \sum_{i=1}^{|V|} a_i x_i \leq b)$ is a knapsack polytope.

4.6 Monotone Submodular Function Maximization

4.6.1 Cardinality and Matroid Constraints: Greedy Approximation

One of the most fundamental results in submodular function maximization is the greedy algorithm presented in [40]. This algorithm can be summarized as:

Algorithm 1 Greedy Submodular Function Maximization subject to the Cardinality Constraint [40, 35]

```
1:  maximize(f, V, k){
2:      S ← φ, U ← V
3:      While(|S| ≤ k){
4:          i* ← argmax_{i∈U} f(S ∪ {i})
5:          If(f(S ∪ {i*}) < f(S))
6:              break
7:          EndIf
8:          S ← S ∪ {i*}, U ← U\{i*}
9:      EndWhile
10:     return S
11: end
```

The authors of [40] prove that this algorithm achieves $1 - 1/e$ approximation for the problem of maximizing a submodular function subject to the cardinality constraint. This result also generalizes to max-k-coverage result. Further, [41] shows that there is no $(1 - 1/e + \epsilon)$ approximation for max-k-coverage problem, unless $P = NP$. The authors of [40], also generalized their result to the case of k-matroids. Specifically, they state that subject to a k-matroid constraint, the greedy algorithm achieves 1/2-approximation for maximization of a submodular function.

4.6.2 Other Techniques

1. **Partial Enumeration:** In this technique, we simply use brute force enumeration over all possible subsets up to size k and then continue with the greedy approach. [42] using partial enumeration, shows a $(1 - 1/e)$ approximation for monotone submodular function maximization subject to the knapsack constraints.

2. **Local Search:** Local search algorithms move from solution to solution among the potential solutions, until a solution does not provide any improvement or the time-limit is passed.

3. **Multilinear Extension:** *Multilinear extension* of a submodular function f is defined as [35]:

$$F(x) = \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i) \tag{4.2}$$

   where x_i is the value of i-th dimension of x. Multilinear extension can also be interpreted as $F(x) = E[f(\hat{x})]$ where $\hat{x}$ is obtained by randomly rounding each x_i to 0 or 1 with probability x_i. Unlike Lovasz extension, multilinear extension is neither convex nor concave. However, a large number of approximation results are achieved by using this extension.

An overview of the results on constrained submodular function maximization using the methods above are summarized in Table 2.

Table 2: Overview of the results on constrained monotone submodular function maximization [28]

Ref.	Constraint	Approximation	Hardness	Technique
[40]	Cardinality	$1-1/e$	$1-1/e$	Greedy
[40]	Matroid	$1-1/e$	$1-1/e$	Multilinear Extension
[42]	$O(1)$ Knapsacks	$1-1/e$	$1-1/e$	Multilinear Extension
[43]	k Matroids	$k+\epsilon$	$k/\log k$	Local Search
[44]	k Matroids & $O(1)$ Knapsacks	$O(k)$	$k/\log k$	Multilinear Extension

4.7 Non-monotone Submodular Function Maximization

4.7.1 Unconstrained Setting

[45], provides a 1/3-approximation algorithm based on local search for the unconstrained non-monotone submodular function maximization. However recently, the authors of [46] proposed a simple randomized linear time algorithm which has a tight 1/2 approximation guarantee. It is important to mention that this result has the same bound pointed out in [45] and it is NP-Hard to do better than this bound. Prior to this, authors also provide a deterministic version of this algorithm which has a 1/3-approximation factor. The algorithm which is called "Randomized USM[1]"is summarized below:

Algorithm 1 Randomized USM [46, 35]

maximize$(f, V)\{$
  $s_0 \leftarrow \emptyset, S_0 \leftarrow V$
  For $i = 1$ *to* $|V|$
    $a_i \leftarrow MAX\left(f(s_i \cup \{i\}) - f(s_i), 0\right)$
    $b_i \leftarrow MAX\left(f(S_i \cup \{i\}) - f(S_i), 0\right)$
    with probability $\frac{a_i}{a_i + b_i}$ **do**
      $s_{i+1} \leftarrow s_i \cup \{i\}, S_{i+1} \leftarrow S_i$
    else
      $s_{i+1} \leftarrow s_i, S_{i+1} \leftarrow S_i \cup \{i\}$
    end

  EndFor
end

Randomized USM has many similarities to Greedy algorithm for cardinality constraint. The table below shows a comparison of these two:

[1]Randomized **U**nconstrained **S**ubmodular **M**aximization

Table 3: Comparison of the greedy algorithm of [40] and randomized algorithm of [46]. (from [35])

Ref.	Method	Radomized	Time	Approximation Guarantee
[40]	Greedily add to $f(\emptyset)$	No	$O(\|V\|^2)$	$1-1/e$ for monotone f
[46]	Greedily add to $f(\emptyset)$ and remove from $f(V)$	Yes	$O(\|V\|)$	$1/2$ for all f

4.7.2 Constrained Setting

The same authors of Randomized USM, in [47], provide a $max\left\{1/e+0.004, \left(1+\frac{|V|}{2\sqrt{(|V|-k)k}}\right)^{-1}-o(1)\right\}$-approximation randomized greedy algorithm for capped cardinality constrained setting (when $|S| \leq k$) of maximization of a non-monotone submodular function. For the case of cardinality fixed setting (when $|S| = k$), combining with another greedy algorithm, they achieve a $max\left\{\frac{1-\frac{k}{e|V|}}{e}-\epsilon, \left(1+\frac{|V|}{2\sqrt{(|V|-k)k}}\right)^{-1}-o(1)\right\}$-approximation algorithm. The combined algorithm simple outputs the best of the first and second algorithm, which explains the form of its approximation factor. It is worth mentioning that there are only two hardness results, which are for the case of $k = |V|/2$ [48] that is $1/2$, and for the case of $k = o(|V|)$ [49] that is 0.491.

As the cardinality constraints can be achieved by restricting S to independent sets of a Matroid (e.g. a uniform matroid for the case of $S \leq k$), we can also put restrictions in terms of polytopes. [44] develop an approximation algorithm where the convex hull of the feasible sets forms a down-monotone, solvable polytope. The approximation guarantees for this problem depend on several basic properties of multilinear extension of the desired function.

Similar to the case of monotone maximization, the techniques mentioned in Section 4.6.2 have also been used for maximizing submodular functions subject to different constraints. Table 4 shows an overview of these improvements.

Table 4: Overview of the results on non-monotone submodular function maximization [28]

Ref.	Constraint	Approximation	Hardness	Technique
[46]	Unconstrained	$1/2$	$1/2$	Greedy
[47]	Capped Cardinality	See section 4.7.2	-	Greedy
[47]	Fixed Cardinality	See section 4.7.2	-	Greedy
[50]	Matroid	$1/e$	0.48	Multilinear Extension
[50]	$O(1)$ Knapsacks	$1/e$	0.49	Multilinear Extension
[44]	k Matroids	$k+O(1)$	$k/\log k$	Local Search
[44]	k Matroids & $O(1)$ Knapsacks	$O(k)$	$k/\log k$	Multilinear Extension

4.8 Possible Future Work

1. Generally, introducing new approximation algorithms for more general constraints, e.g. online and adaptive extensions are of high interest. [51] is one of the most recent works in this field

and the authors develop a $1/2-\epsilon$ approximation algorithm to the optimum solution, requiring only a single pass through the data, and memory independent of data size.

2. As many cases of submodularity appear in large-scale data, handling large data whether by approximation, randomization or distributed implementation is also a promising research area. [52] proposes a distributed scheme for submodular function maximization that can be easily implemented in a MapReduce framework.

References

[1] M. Gomez Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10*. New York, New York, USA: ACM Press, Jul. 2010, p. 1019.

[2] P. Kohli, L. Ladický, and P. H. S. Torr, "Robust higher order potentials for enforcing label consistency," *International Journal of Computer Vision*, vol. 82, pp. 302–324, 2009.

[3] A. Krause and D. Golovin, "Submodular function maximization," *Tractability: Practical Approaches to Hard Problems*, vol. 3, pp. 71–104, 2014.

[4] S. Iwata, "Submodular Functions: Optimization and Approximation," *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) - Vol. I: Plenary Lectures and Ceremonies, Vols. II-IV: Invited Lectures*, pp. 2943–2963, 2010.

[5] L. Lovasz, "Submodular functions and convexity," in *11th International Symposium Mathematical programming, Bonn*, 1982, pp. 235–257.

[6] S. Iwata, "Submodular function minimization," *Mathematical Programming, Ser. B*, vol. 112, pp. 45–64, 2008.

[7] J. Edmonds, "Submodular functions, matroids, and certain polyhedra," *Calgary International Conference on Combinatorial Structures and Their Applications*, pp. 69–87, 1970.

[8] A. Krause and S. Jegelka, "ICML Tutorial: Submodularity in Machine Learning - New Directions," *http://www.cs.berkeley.edu/\tilde stefje/submodularity_icml.html*, 2013.

[9] V. Cevher and A. Krause, "Greedy Dictionary Selection for Sparse Representation," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, no. 5, pp. 979–988, Sep. 2011.

[10] M. Grötschel, L. Lovász, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, pp. 169–197, 1981.

[11] ——, *Geometric Algorithms and Combinatorial Optimization*, ser. Algorithms and Combinatorics. Springer, 1988, vol. 2.

[12] L. G. Khachiyan, "A polynomial algorithm in linear programming," *Doklady Akademii Nauk SSSR*, vol. 244, pp. 1093–1096, 1979.

[13] P. Stobbe and A. Krause, "Efficient Minimization of Decomposable Submodular Functions," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 2208–2216.

[14] S. Fujishige and S. Isotani, "A submodular function minimization algorithm based on the minimum-norm base," *Pacific Journal of Optimization*, vol. 7, pp. 3–17, 2011.

[15] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[16] W. H. Cunningham, "Testing membership in matroid polyhedra," pp. 161–188, 1984.

[17] ——, "On submodular function minimization," *Combinatorica*, vol. 5, pp. 185–192, 1985.

[18] S. Iwata, L. Fleischer, and S. Fujishige, "A Combinatorial Strongly Polynomial Algorithm for Minimizing Submodular Functions," *J. ACM*, vol. 48, no. 4, pp. 761–777, Jul. 2001.

[19] A. Schrijver, "A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time," *J. Comb. Theory Ser. B*, vol. 80, no. 2, pp. 346–355, Nov. 2000.

[20] L. Fleischer and S. Iwata, "Improved Algorithms for Submodular Function Minimization and Submodular Flow," in *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, ser. STOC '00. New York, NY, USA: ACM, 2000, pp. 107–116.

[21] ——, "A Push-relabel Framework for Submodular Function Minimization and Applications to Parametric Optimization," *Discrete Appl. Math.*, vol. 131, no. 2, pp. 311–322, Sep. 2003.

[22] S. Iwata, "A Faster Scaling Algorithm for Minimizing Submodular Functions," in *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization*. London, UK, UK: Springer-Verlag, 2002, pp. 1–8.

[23] J. B. Orlin, "A faster strongly polynomial time algorithm for submodular function minimization." *Math. Program.*, vol. 118, no. 2, pp. 237–251, 2009.

[24] S. Iwata and K. Nagano, "Submodular function minimization under covering constraints," in *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2009, pp. 671–680.

[25] J. Bilmes, "Submodular functions, their optimization and applications," *http://melodi.ee.washington.edu/\tilde bilmes/ee595a_spring_2011/lecture20.pdf*, 2011.

[26] M. X. Goemans and V. S. Ramakrishnan, "Minimizing submodular functions over families of sets," *Combinatorica*, vol. 15, pp. 499–513, 1995.

[27] M. X. Goemans and J. A. Soto, "Symmetric Submodular Function Minimization Under Hereditary Family Constraints," *http://arxiv.org/abs/1007.2140*, Jul. 2010.

[28] J. Vondrak, "Tutorial on submodular optimization," *http://theory.stanford.edu/\tilde jvondrak/data/submod-tutorial-1.pdf*, 2012.

[29] A. Delong, O. Veksler, A. Osokin, and Y. Boykov, "Minimizing Sparse High-Order Energies by Submodular Vertex-Cover," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 962–970.

[30] S. Sachdeva and R. Saket, "Nearly Optimal NP-Hardness of Vertex Cover on k-Uniform k-Partite Hypergraphs," *http://arxiv.org/abs/1105.4175*, May 2011.

[31] C. Chekuri and A. Ene, "Approximation Algorithms for Submodular Multiway Partition," in *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, Oct. 2011, pp. 807–816.

[32] F. A. Chudak and K. Nagano, "Efficient Solutions to Relaxations of Combinatorial Problems with Submodular Penalties via the Lov\ÁSz Extension and Non-smooth Convex Optimization," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 79–88.

[33] G. Goel, C. Karande, P. Tripathi, and L. Wang, "Approximability of combinatorial problems with multi-agent submodular cost functions," in *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2009, pp. 755–764.

[34] S. Jegelka and J. Bilmes, "Online Submodular Minimization for Combinatorial Structures," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ser. ICML '11, L. Getoor and T. Scheffer, Eds. New York, NY, USA: ACM, Jun. 2011, pp. 345–352.

[35] J. Gillenwater, "Maximization of Non-Monotone Submodular Functions," *http://www.seas.upenn.edu/\tilde jengi/wpeii2014.pdf*, 2014.

[36] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*, 2003, p. 137.

[37] G. L. Nemhauser and L. A. Wolsey, "Maximizing submodular set functions: formulations and analysis of algorithms," Tech. Rep., 1981.

[38] Y. Kawahara and J. A. Bilmes, "Submodularity Cuts and Applications," in *Neural Information Processing Systems*, 2009, pp. 916—-924.

[39] B. Goldengorin, G. Sierksma, G. A. Tijssen, and M. Tso, "The Data-Correcting Algorithm for the Minimization of Supermodular Functions," pp. 1539–1551, 1999.

[40] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, pp. 265–294, 1978.

[41] U. Feige, "A threshold of ln n for approximating set cover," pp. 634–652, 1998.

[42] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, Jan. 2004.

[43] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko, "Non-monotone submodular maximization under matroid and knapsack constraints," in *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC '09*. New York, New York, USA: ACM Press, May 2009, p. 323.

[44] C. Chekuri, J. Vondrák, and R. Zenklusen, "Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes," *http://arxiv.org/abs/1105.4593*, May 2011.

[45] U. Feige, V. S. Mirrokni, and J. Vondrák, "Maximizing non-monotone submodular functions," in *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2007, pp. 461–471.

[46] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz, "A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization," in *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, Oct. 2012, pp. 649–658.

[47] N. Buchbinder, M. Feldman, J. S. Naor, and R. Schwartz, "Submodular Maximization with Cardinality Constraints," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '14. SIAM, 2014, pp. 1433–1452.

[48] J. Vondrák, "Symmetry and approximability of submodular maximization problems," in *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2009, pp. 651–670.

[49] S. O. Gharan and J. Vondrák, "Submodular Maximization by Simulated Annealing," *http://arxiv.org/abs/1007.1632*, Jul. 2010.

[50] S. Oveis Gharan and J. Vondrák, "On variants of the matroid secretary problem," in *Algorithmica*, vol. 67, 2013, pp. 472–497.

[51] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause, "Streaming Submodular Maximization: Massive Data Summarization on the Fly," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14. New York, NY, USA: ACM, 2014, pp. 671–680.

[52] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed Submodular Maximization: Identifying Representative Elements in Massive Data," in *Advances in Neural Information Processing Systems*, 2013, pp. 2049–2057.